

Proceedings of the 15th Conference on Natural Language Processing

(KONVENS 2019)

October 9 – 11, 2019

German Society for Computational Linguistics & Language Technology
Friedrich-Alexander-Universität Erlangen-Nürnberg
Edited by the Chair of Computational Corpus Linguistics

Contents

Introduction	iii
Program Committee	iii
Acknowledgements	iv
Organizers	iv
KONVENS 2019 Papers	1
Long Papers	1
BERT for Named Entity Recognition in Contemporary and Historic German <i>Kai Labusch, Clemens Neudecker and David Zellhöfer</i>	1
A Supervised Learning Approach for the Extraction of Sources and Targets from German Text <i>Michael Wiegand, Margarita Chikobava and Josef Ruppenhofer</i>	10
A Descriptive Analysis of a German Corpus Annotated with Opinion Sources and Targets <i>Michael Wiegand, Leonie Lapp and Josef Ruppenhofer</i>	20
Automated Assessment of Language Proficiency on German Data <i>Edit Szügyi, Sören Etler, Andrew Beaton and Manfred Stede</i>	30
A Probabilistic Morphology Model for German Lemmatization <i>Christian Wartena</i>	40
To Act Or Not To Act - Annotating and Classifying Email Regarding Necessary Action <i>Veronika Hintzen and Alexander Fraser</i>	50
AkkuBohrHammer vs. AkkuBohrhammer: Experiments towards the Evaluation of Compound Splitting Tools for General Language and Specific Domains <i>Anna Häfty, Ulrich Heid, Anna Moskvina, Julia Bettinger, Michael Dorna and Sabine Schulte im Walde</i>	59
Neural classification with attention assessment of the implicit-association test OMT and prediction of subsequent academic success <i>Dirk Johannßen and Chris Biemann</i>	68
Towards Multimodal Emotion Recognition in German Speech Events in Cars using Transfer Learning <i>Deniz Cevher, Sebastian Zepf and Roman Klinger</i>	79
Visualising and evaluating the effects of combining active learning with word embedding features <i>Maria Skeppstedt, Rafal Rzepka, Kenji Araki and Andreas Kerren</i>	91
Creating Information-maximizing Natural Language Messages Through Image Captioning-Retrieval <i>Fabian Karl, Mikko Lauri and Chris Biemann</i>	101

German End-to-end Speech Recognition based on DeepSpeech <i>Aashish Agarwal and Torsten Zesch</i>	111
Label Propagation of Polarity Lexica on Word Vectors <i>Harald Koppen and Ritavan</i>	120
Detecting the boundaries of sentence-like units in spoken German <i>Josef Ruppenhofer and Ines Rehbein</i>	130
Dependency Trees for Greenlandic <i>Eckhard Bick</i>	140
Metaphor detection for German Poetry <i>Ines Reinig and Ines Rehbein</i>	149
Does BERT Make Any Sense? Interpretable Word Sense Disambiguation with Contextualized Embeddings <i>Gregor Wiedemann, Steffen Remus, Avi Chawla and Chris Biemann</i>	161
Determining Response-generating Contexts on Microblogging Platforms <i>Jennifer Fest, Arndt Heilmann, Oliver Hohlfeld, Stella Neumann, Jens Helge Reelfs, Marco Schmitt and Alina Vogelgesang</i>	171
Extraction and Classification of Speech, Thought, and Writing in German Narrative Texts <i>Luise Schricker, Manfred Stede and Peer Trilcke</i>	183
Argumentative Relation Classification as Plausibility Ranking <i>Juri Opitz</i>	193
Predicting Semantic Labels of Text Regions in Heterogeneous Document Images <i>Somtochukwu Enendu, Johannes Scholtes, Jeroen Smeets, Djoerd Hiemstra and Mariet Theune</i>	203
Evaluating Off-the-Shelf NLP Tools for German <i>Katrin Ortmann, Adam Roussel and Stefanie Dipper</i>	212
Short Papers	223
Towards a gold standard corpus for detecting valencies of Zulu verbs <i>Gertrud Faqß and Sonja Bosch</i>	223
“Konservenglück in Tiefkühl-Town” – Das Songkorpus als empirische Ressource interdisziplinärer Erforschung deutschsprachiger Poptexte <i>Roman Schneider</i>	229
Combining embedding methods for a word intrusion task <i>Finn Årup Nielsen and Lars Kai Hansen</i>	237
Deep learning for Free Indirect Representation <i>Annelen Brunner, Ngoc Duyen Tanja Tu, Lukas Weimer and Fotis Jannidis</i>	241
Representing document-level semantics of biomedical literature using pre-trained embedding models: Novel assessments <i>Jon Stevens, Brandon Punturo, Derek Chen, Mike Kim and Jacob Zimmer</i>	246

Deep-EOS: General-Purpose Neural Networks for Sentence Boundary Detection <i>Stefan Schweter and Sajawel Ahmed</i>	251
How Does Visual Complexity Influence Predictive Language Processing in a Situated Context? <i>Özge Alacam, Wolfgang Menzel and Tobias Staron</i>	256
Teacher-Student Learning Paradigm for Tri-training: An Efficient Method for Unlabeled Data Exploitation <i>Yash Bhalgat, Zhe Liu, Pritam Gundecha, Jalal Mahmud and Amita Misra</i>	262
Kaleidoscope Abstracts	267
Generic Web Content Extraction with Open-Source Software <i>Adrien Barbaresi</i>	267
Ein Tool zur Visualisierung des Gebrauchs von Schreibvarianten <i>Peter M. Fischer and Christian Lang</i>	269
Identification of Reading Absorption in User-Generated Book Reviews <i>Piroska Lendvai, Simone Rebora and Moniek Kuijpers</i>	271
Sketches of a graphical user interface for word alignment annotation <i>Maria Skeppstedt, Magnus Ahltorp, Gunnar Eriksson and Rickard Domeij</i>	273
Predicting default and non-default aspectual coding: Impact and density of information features <i>Michael Richter and Tariq Yousef</i>	275
A New German Reddit Corpus <i>Andreas Blombach, Natalie Dykes, Stefan Evert, Philipp Heinrich, Besim Kabashi and Thomas Proisl</i>	278
GermEval 2019 Papers	280
Task 1 Papers	280
GermEval 2019 Task 1: Hierarchical Classification of Blurbs <i>Steffen Remus, Rami Aly and Chris Biemann</i>	280
Multi-Label Classification of Blurbs with SVM Classifier Chains <i>Franz Bellmann, Lea Bunzel, Christoph Demus, Lisa Fellendorf, Olivia Gräupner, Qiuyi Hu, Tamara Lange, Alica Stuhr, Jian Xi, Dirk Labudde and Michael Spranger</i>	293
Multi-Label Multi-Class Hierarchical Classification using Convolutional Seq2Seq <i>Venkatesh Umaashankar and Girish Shanmugam S</i>	300
Enriching BERT with Knowledge Graph Embeddings for Document Classifica- tion <i>Malte Ostendorff, Peter Bourgonje, Maria Berger, Julian Moreno-Schneider, Georg Rehm and Bela Gipp</i>	307

COMTRAVO-DS team at GermEval 2019 Task 1 on Hierarchical Classification of Blurbs	
<i>David S. Batista and Matti Lyra</i>	315
Convolutional Neural Networks for Classification of German Blurbs	
<i>Erdan Genc, Louay Abdelgawad, Viorel Morari and Peter Kluegl</i>	323
TwistBytes - Hierarchical Classification at GermEval 2019: walking the fine line (of recall and precision)	
<i>Fernando Benites</i>	328
The HUIU Contribution to the GermEval 2019 Shared Task 1	
<i>Melanie Andresen, Melitta Gillmann, Jowita Grala, Sarah Jablotschkin, Lea Roseler, Eleonore Schmitt, Lena Schnee, Katharina Straka, Michael Vauth, Sandra Kübler, Heike Zinsmeister</i>	336
Label Frequency Transformation for Multi-Label Multi-Class Text Classification	
<i>Raghavan A K, Venkatesh Umaashankar and Gautham Krishna Gudur</i>	341
Logistic Regression and Naive Bayes for Hierarchical Multi-label Classification at GermEval 2019 - Task 1	
<i>Kristian Rother and Achim Rettberg</i>	347
Task 2 Papers	354
Overview of GermEval Task 2, 2019 Shared Task on the Identification of Offensive Language	
<i>Julia Maria Struß, Melanie Siegel, Josep Ruppenhofer, Michael Wiegand and Manfred Klenner</i>	354
InriaFBK Drawing Attention to Offensive Language at Germeval2019	
<i>Michele Corazza, Stefano Menini, Elena Cabrio, Sara Tonelli and Serena Villata</i>	366
German Hatespeech classification with Naive Bayes and Logistic Regression - hshl at GermEval 2019 - Task 2	
<i>Kristian Rother and Achim Rettberg</i>	372
FraunhoferSIT at GermEval 2019: Can Machines Distinguish Between Offensive Language and Hate Speech? Towards a Fine-Grained Classification	
<i>Inna Vogel and Roey Regev</i>	377
FoSIL - Offensive language classification of German tweets combining SVMs and deep learning techniques	
<i>Florian Schmid, Justine Thielemann, Anna Mantwill, Jian Xi, Dirk Labudde and Michael Spranger</i>	382
2019 GermEval Shared Task on Offensive Tweet Detection h_da submission	
<i>Isabell Börner, Midhad Blazevic, Maximilian Komander and Margot Mieskes</i>	387

HAU at the GermEval 2019 Shared Task on the Identification of Offensive Language in Microposts: System Description of Word List, Statistical and Hybrid Approaches	
<i>Johannes Schäfer, Tom De Smedt and Sylvia Jaki</i>	391
UPB at GermEval-2019 Task 2: BERT-Based Offensive Language Classification of German Tweets	
<i>Andrei Paraschiv and Dumitru-Clementin Cercel</i>	398
hpiDEDIS at GermEval 2019: Offensive Language Identification using a German BERT model	
<i>Julian Risch, Anke Stoll, Marc Ziegele and Ralf Krestel</i>	405
The HUIU Contribution for the GermEval Shared Task 2	
<i>Melanie Andresen, Melitta Gillmann, Jowita Grala, Sarah Jablotschkin, Lea Röseler, Eleonore Schmitt, Lena Schnee, Katharina Straka, Michael Vauth, Sandra Kübler and Heike Zinsmeister</i>	411
TUWienKBS19 at GermEval Task 2, 2019: Ensemble Learning for German Offensive Language Detection	
<i>Joaquín Padilla Montani and Peter Schüller</i>	418
RGCL at GermEval 2019: Offensive Language Detection with Deep Learning	
<i>Alistair Plum, Tharindu Ranasinghe, Constantin Orăsan, Ruslan Mitkov</i>	423
FKIE - Offensive Language Detection on Twitter at GermEval 2019	
<i>Theresa Krumbiegel</i>	429
bertZH at GermEval 2019: Fine-Grained Classification of German Offensive Language using Fine-Tuned BERT	
<i>Tim Graf and Luca Salini</i>	434

Introduction

The Conference on Natural Language Processing (“Konferenz zur Verarbeitung natürlicher Sprache”, KONVENS) aims at offering a broad perspective on current research and developments within the interdisciplinary field of natural language processing. It allows researchers from all disciplines relevant to this field of research to present their work.

The 15th KONVENS “Bridging the gap between NLP and human understanding” took place October 9–11, 2019, at Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany. It was the first KONVENS in an annual schedule, following KONVENS 2018 in Vienna, Austria, and preceding KONVENS 2020 in Zürich, Switzerland. KONVENS 2019 was collocated with the GermEval Workshop and a Statistics Tutorial.

The conference was organized by the Chair of Computational Corpus Linguistics led by Prof. Dr. Stefan Evert.

Program Committee

Barbaresi, Adrien	Pado, Sebastian
Biemann, Chris	Pinkal, Manfred
Bollmann, Marcel	Pirker, Hannes
Buchberger, Ernst	Quasthoff, Uwe
Butt, Miriam	Rehbein, Ines
Clematide, Simon	Rehm, Georg
Crysmann, Berthold	Ruppenhofer, Josef
De William, Ernesto Luca	Sasaki, Felix
Dipper, Stefanie	Schäfer, Roland
Heid, Ulrich	Scherrer, Yves
Heiden, Serge	Schmid, Helmut
Klenner, Manfred	Schmidt, Thomas
Klinger, Roman	Schulte im Walde, Sabine
Kordoni, Valia	Stede, Manfred
Krenn, Brigitte	Tanguy, Ludovic
Kübler, Sandra	Versley, Yannick
Lapshinova-Koltunski, Ekaterina	Weskott, Thomas
Mehler, Alexander	Wolska, Magdalena
Menzel, Wolfgang	Zesch, Torsten
Nakov, Preslav	Zinsmeister, Heike
Neumann, Günter	

Acknowledgements

Invited speakers:

Gemma Boleda (Universitat Pompeu Fabra, Barcelona): Computational linguistics and linguistic theory

Daisuke Bekki (Ochanomizu University) and Hitomi Yanaka (RIKEN Center for Advanced Intelligence Project): Hybrid natural language understanding: neural network, logic and beyond

Special thanks:

Prof. Dr. Günther Görz, FAU Erlangen-Nürnberg

Prof. Dr. Joachim Hornegger, President of FAU Erlangen-Nürnberg

Prof. Dr. Rainer Trinczek, Dean of the Faculty of Humanities, Social Sciences, and Theology

Financial support:

Dr. German Schweiger-Stiftung

Luise Prell Stiftung

Gastprofessorenprogramm der FAU, gefördert durch das Bayerische Staatsministerium für Wissenschaft und Kunst

Organizers

Blombach, Andreas

Dykes, Natalie

Evert, Stefan

Greiner, Paul

Griebel, Tim

Heinrich, Philipp

Kabashi, Besim

Proisl, Thomas

Schorr, Tanja

BERT for Named Entity Recognition in Contemporary and Historical German

Kai Labusch

Staatsbibliothek zu Berlin -
Preußischer Kulturbesitz
10785 Berlin, Germany
kai.labusch
@sbb.spk-berlin.de

Clemens Neudecker

Staatsbibliothek zu Berlin -
Preußischer Kulturbesitz
10785 Berlin, Germany
clemens.neudecker
@sbb.spk-berlin.de

David Zellhoefer

Staatsbibliothek zu Berlin -
Preußischer Kulturbesitz
10785 Berlin, Germany
david.zellhoefer
@sbb.spk-berlin.de

Abstract

We apply a pre-trained transformer based representational language model, i.e. BERT (Devlin et al., 2018), to named entity recognition (NER) in contemporary and historical German text and observe state of the art performance for both text categories. We further improve the recognition performance for historical German by unsupervised pre-training on a large corpus of historical German texts of the Berlin State Library and show that best performance for historical German is obtained by unsupervised pre-training on historical German plus supervised pre-training with contemporary NER ground-truth.

1 Introduction

The transformer (Vaswani et al., 2017) is a recent neural network architecture that has been used as the central building block of representational language models such as GPT (Radford et al., 2018) or BERT (Devlin et al., 2018). These representational models can either be utilized to derive features that serve as input for other models such as a long short term memory (LSTM) and/or a conditional random field (CRF) or they can be directly trained on some supervised task. In this paper, we follow the latter approach and train a pre-trained BERT model directly for named entity recognition (NER) tasks.

In contrast to contemporary German, historical German texts pose multiple challenges on a potential algorithm because their language is less standardized and their digital representation has been typically obtained by optical character recognition (OCR) that has been shown to be error prone in this particular scenario (Federbusch et al., 2013).

In the experiments presented below, we evaluate the performance of BERT on two contemporary German NER data sets as well as on three different

historical German NER corpora (see Sec. 5). We get best results for historical German by application of unsupervised pre-training on a large historic german text corpus plus supervised pre-training using contemporary German NER ground-truth. In contrast best results for contemporary German are obtained without unsupervised pre-training. The large historical German text corpus that is used for unsupervised pre-training has been extracted from the digital collections of the Berlin State Library (Staatsbibliothek zu Berlin/SBB).

The software used in the experiments is provided for download ¹.

2 Background

The SBB is digitizing its copyright-free holdings and makes them publicly available online in various formats for direct² or automated³ download. As part of an on-going process, a growing amount of OCR-derived full-texts of the digitized printed material is provided in ALTO⁴ format but is mainly used for internal use cases such as full-text indexing and other information retrieval tasks.

However, OCR of historic documents is significantly more difficult than OCR of modern texts due to the large variety of fonts, layouts, mixed languages, and non-standardized orthography of printed texts from before 1850. As a consequence, texts generated by standard OCR contain a high amount of word errors. Similar challenges have been described by (Lopresti, 2009) and (Alex and Burns, 2014) who have noted that the quality of text analysis is directly tied to the level of noise in a document. Additional difficulties are caused by the historic language (Piotrowski, 2012).

Despite these obstacles, natural language processing – and NER in particular – strongly con-

¹https://github.com/qurator-spk/sbb_ner

²<https://digital.staatsbibliothek-berlin.de>

³<https://digital.staatsbibliothek-berlin.de/doi>

⁴<https://www.loc.gov/standards/alto/>

tribute to an improvement of the user experience as they leverage supportive means for exploration and search within large text corpora. Furthermore, a growing research interest from the Digital Humanities in text and, e.g., data mining for historical social network analysis relies on the extraction of named entities from the digitized and OCR-derived full-text collections.

First experiences with NER for historical texts at the SBB were obtained in the Europeana Newspapers project where a CRF (Finkel et al., 2005) was trained on manually labeled OCR texts of historic newspapers (Neudecker et al., 2014). This approach was superseded in the Oceanic Exchanges project where (Riedl and Padó, 2018) achieve state of the art results for historic German by combining a bidirectional long short term memory (biLSTM) with a CRF as top layer and transfer learning.

The work presented in this paper aims towards a versatile approach that performs decently on texts of different epochs, i.e. contemporary and historical, without requirement of intense parameter tuning with respect to particular target corpora.

The paper is structured as follows: The next section outlines the relevant work in the context of the presented approach. Section 4 describes four data sets that are used in the three experiments presented. In particular, it presents the data of the Berlin State Library that has not been published so far. Then, Section 5 gives a brief description of the technical details of the experiments. The outcome of the experiments is discussed and interpreted in Section 6. The paper concludes with an outlook on future work.

3 Related Work

(Grover et al., 2008) designed a rule-based system for recognizing person and place names in digitized records of British parliamentary proceedings from the late 17th and early 19th centuries and report F_1 -scores from 70.35 to 76.94 percent.

(Packer et al., 2010) compare the performance of a dictionary-based extractor, a regular expression rule-based extractor, a Maximum Entropy Markov Model (MEMM) and a CRF on historical OCR-processed documents with a mean word error rate of 56 percent, revealing that a voting-based ensemble method can boost F_1 -scores from 60.7 to 68 percent.

For a corpus of historic French newspapers, (Gal-

ibert et al., 2012) report F_1 -scores between 55.2 and 68.9 percent for two stochastic and one rule-based system by including noisy entities in the annotations.

In the Europeana Newspapers project, (Neudecker et al., 2014) measure F_1 -scores of 46.6 to 73.27 percent with a CRF trained on annotated noisy OCR from historic newspapers in Dutch, French, and German. F_1 -scores up to 60 percent are obtained for a dataset of Finnish OCR-treated newspapers from the 19th and early 20th century with a rule-based system (Kettunen et al., 2016) and the Finnish Semantic Tagger, a lexicon-based semantic tagger (Kettunen and Ruokolainen, 2017).

A supervised machine learning system (Nouvel et al., 2011) has been shown to improve F_1 -score up to 76.1 percent (Ehrmann et al., 2016). This result was improved furthermore by (Riedl and Padó, 2018) where transfer learning from the German Europeana Newspapers data enabled the biLSTM+CRF classifier to reach a top F_1 -score of 78.56 percent (see Table 2).

To conclude, (Schweter and Baiter, 2019) recently employed BERT features for NER resulting in F_1 -scores from 75.31 to 79.14 percent while their best models that have been trained on newspaper data of corresponding time epochs deliver F_1 -scores from 77.51 to 85.32 percent (see also Table 3).

4 Datasets

4.1 Europeana Newspapers Historic German Datasets

The Europeana Newspapers NER corpus was derived from historical newspapers that have been processed by an OCR and subsequently annotated (Neudecker, 2016). Therefore, that corpus constitutes a good match for the kind of material addressed in this paper. It comprises data sets for historical Dutch, French, and German where the German data has been sourced from newspapers from 1926 from the Dr Friedrich Tessmann Library (LFT), newspapers from 1710 to 1873 from the Austrian National Library (ONB), and newspapers from 1872 to 1930 from the Berlin State Library (SBB).

4.2 CoNLL 2003 German Named Entity Recognition Ground Truth

The German data used in the CoNLL 2003 task (Tjong Kim Sang and De Meulder, 2003) has been taken from a German newspaper, the Frankfurter Rundschau, from 1992. The CoNLL set possesses two different test sets, i.e. TEST-A and TEST-B. We use both in the experiments only for testing (DE-CoNLL-TEST).

4.3 GermEval Konvens 2014 Shared Task Data

The GermEval dataset (Benikova et al., 2014) has been sourced from sampling German Wikipedia and various online newspapers. The GermEval dataset possesses a training, a development and a test set. The development set has not been used at all in the experiments.

4.4 Distribution of Entities

The distribution of labeled entity tokens within the different NER ground truth data sets is shown in Table 1.

	LOC	ORG	PER	Size
DE-CoNLL-TEST	0.025	0.033	0.037	103387
DE-CoNLL-TRAIN	0.025	0.020	0.022	206931
GermEval-TEST	0.028	0.021	0.027	96499
GermEval-TRAIN	0.028	0.022	0.027	452853
LFT	0.062	0.037	0.067	70259
ONB	0.066	0.007	0.115	28012
SBB	0.022	0.010	0.019	47281

Table 1: Distribution of entity tokens amongst different training sets and frequencies of entity tokens across different training sets.

4.5 Digital Collections of the Berlin State Library (DC-SBB)

At the time of the writing of this paper, the digital collections of the SBB contain 153,942 digitized works from the time period of 1470 to 1945 (see Figure 1). Up to now, 28,909 works have been OCR-processed resulting in 4,988,099 full-text pages.

We applied a sequence of filter steps in order to exclude pages that do not contain german text, have very bad OCR results or contain content that is unlikely to be continuous text.

For each page with OCR text, we predicted its language by means of the *langid* tool (Lui and Baldwin, 2012). Figure 2 illustrates the number of pages per language limited to the most frequent

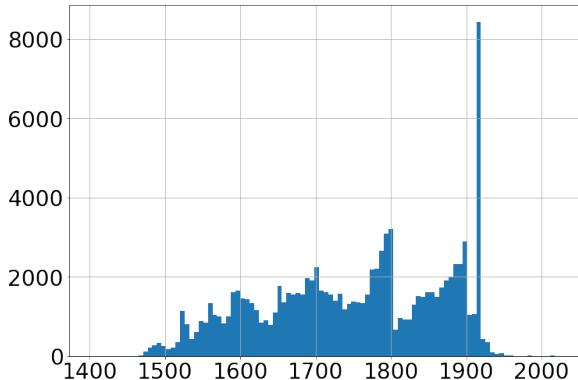


Figure 1: Distribution of publication dates in the digital collections of the Berlin State Library (DC-SBB).

languages. For 19,669 works, the language is consistent over all pages as can be seen from the histogram of detected languages per work that is given in Figure 3. Due to this consistency for the vast majority of all works, we consider the per page language detection provided by langid as sufficiently reliable means to filter out non-german pages. Additionally, we take into account only pages with a confidence score of the German language detection greater than 0.999999.

Fulltexts of pages where the OCR did not work at all, for instance pages that contain hand-written parts, tend to look like random character sequences. In order to exclude these “broken” pages from the data, we computed the distribution of the per-page character entropy rate over all pages. Figure 4 depicts the distribution of the per page character entropy rate in the DC-SBB. We excluded all pages with a character entropy rate below the 0.2 percentile or above the 0.8 percentile of that distribution from the dataset.

As a consequence of these filter steps, 2,333,647 pages of unlabeled historical German text remain and form the DC-SBB dataset. The full dataset is available freely online (Labusch and Zellhöfer, 2019).

5 Experiments

In the scope of the three presented experiments, the BERT model is trained directly with respect to the NER by implementation of the same method that has been proposed by the BERT authors (Devlin et al., 2018). During training, the maximum sequence length is set to 128.

Throughout all experiments, we use the Adam optimizer algorithm with decoupled weight decay (Loshchilov and Hutter, 2019) where the weight

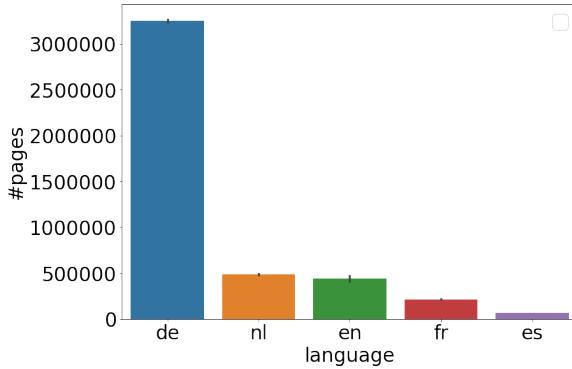


Figure 2: Number of pages per language as detected by *langid* for the most common languages.

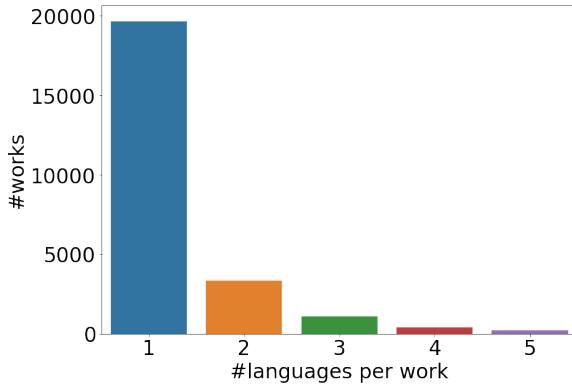


Figure 3: Number of detected languages by *langid* per DC-SBB document (documents with >5 languages are omitted).

decay is set to 0.03. We apply a linear learning rate schedule where warm-up and cool-down of the learning rate take 40% of the performed training steps. We set the target learning rate to $3 * 10^{-5}$ and use a batch size of 32 during all the experiments. We carried out 7 training epochs if not noted otherwise.

Accumulative gradient descent for both supervised and unsupervised learning is applied due to hardware limitations that would otherwise enforce a smaller batch size. Instead of the original BERT implementation, all experimental runs rely on an equivalent PyTorch implementation provided by (Hugging Face, 2019) since accumulative gradient descent cannot be easily carried out using the current Tensorflow (< 2.0) implementation of BERT.

5.1 BERT-Base Multi-Lingual Cased Model

In the first batch of experiments, we explore the NER performance of the baseline model as it has been provided by Google⁵. We use their BERT-

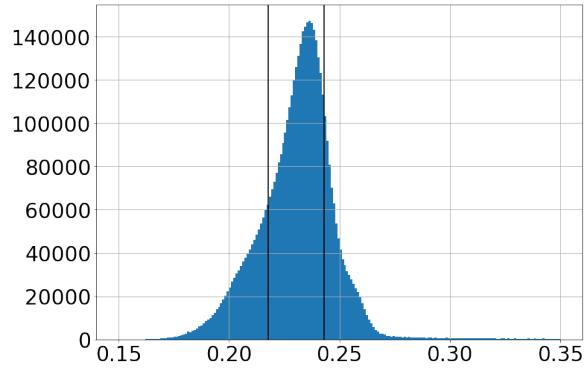


Figure 4: Distribution of the per page character entropy rate of the documents in the DC-SBB dataset. The 0.2 and 0.8 percentiles have been marked with a vertical line.

Base multi-lingual cased model that has been pre-trained on 104 languages. It has 12 transformer blocks where each transformer block has 768 layers with 12 attention heads and uses a vocabulary size of 119,547. The entire model has about 110 million parameters. The left F_1 -column of Table 2 shows the results of the BERT-Base model for different combinations of training and test sets.

5.2 BERT-Base Model with Pre-Training on DC-SBB

In this experimental run, we study the impact of unsupervised pre-training with respect to the NER performance on historical and contemporary data. Therefore the multi-lingual BERT-Base model is pre-trained unsupervisedly on the DC-SBB dataset (see Sec. 4.5). The unsupervised pre-training task is composed of the ‘‘Masked-LM’’ and ‘‘Next Sentence Prediction’’ tasks that have been proposed in (Devlin et al., 2018).

The pre-training of the base model has been run for approximately 500 hours on a single NVIDIA 2080 GPU which is equivalent to 5 epochs. During pre-training, the batch size is set to 128, the learning rate is set as in the NER task training and a weight decay of 0.01 is used. The middle F_1 -column of Table 2 shows results of the BERT-Base model being pre-trained on the DC-SBB data for different combinations of training and test sets.

5.3 5-fold Cross Validation and Comparison with State of the Art Approaches

Since the NER performance varies heavily for different train/test set combinations and in order to make our results comparable to results in (Riedl and Padó, 2018) and (Schweter and Baiter, 2019), we run a third batch of experiments where a 5-fold

⁵<https://github.com/google-research/bert/blob/master/multilingual.md>

cross validation is performed for the three historical German corpora.

In this run, the impact of pre-training on the model performance under cross validation is evaluated. We apply unsupervised pre-training on the DC-SBB data as well as supervised pre-training on contemporary NER ground truth. In case of supervised pre-training, 7 training epochs are run again with the same learning parameters described above. Finally, unsupervised and supervised pre-training are combined where unsupervised is done first and supervised second. The corresponding cross validation results are shown in Table 3.

6 Discussion

The NER ground truth sets that have been used in the experiments described above are diverse in terms of size and with respect to the frequencies of the entity classes as Table 1 summarizes.

While the contemporary data sets GermEval and CoNLL show similar frequencies of entity classes, the frequencies of entities within the historical data sets LFT, ONB, and SBB deviate significantly. The SBB set comes closest to the contemporary sets in terms of entity frequencies.

Futhermore, there is far more contemporary ground truth available than for historical texts. The amount of ground truth also varies significantly among the various historical datasets.

Table 2 shows the NER performance in terms of the F_1 -score obtained with different training/test combinations using either the original BERT-Base model or a BERT-Base model that has been pre-trained on the DC-SBB set. (Riedl and Padó, 2018) present a comprehensive evaluation of CRF and bidirectional long short term memory (biLSTM) with CRF layer approaches for NER in contemporary and historical German, relying on a partial utilization of the ground truth data that is considered in this work. The authors use character embeddings together with different pre-trained word embeddings as input features of the biLSTMs. For those training/test pairs that have corresponding results in (Riedl and Padó, 2018), their best result is listed in the rightmost F_1 -column of Table 2.

Interestingly, unsupervised pre-training on DC-SBB data worsens BERT performance in the case of contemporary training/test pairs while the performance improves for all experiments that test on historical ground truth with one exception (CoNLL/LFT). Please note that the same training

			BERT multi-lingual-cased	(Riedl and Padó, 2018)	
		pre-train:	none	DC-SBB	none
train	test		F_1	F_1	F_1
CoNLL	CoNLL	84.5	82.6	82.99	
	LFT	52.9	52.0	49.28	
	ONB	56.1	56.6	58.79	
	SBB	67.6	68.3	-	
GermEval	GermEval	88.6	86.7	82.93	
	LFT	54.2	54.8	55.99	
	ONB	60.0	62.6	61.35	
	SBB	63.1	65.1	-	
GermEval + CoNLL	CoNLL	80.2	79.4	-	
	GermEval	88.0	85.7	-	
	LFT	55.1	55.2	-	
	ONB	58.6	60.1	-	
LFT	SBB	64.1	65.1	-	
	ONB	71.5	75.9	65.53	
	SBB	54.4	56.9	-	
	ONB	72.5	75.7	-	
LFT+SBB	LFT	59.4	61.5	49.35	
	SBB	51.3	54.6	-	
	ONB	54.0	55.5	-	
	LFT	61.9	62.7	-	
ONB+LFT	SBB	53.9	54.9	-	
	ONB+SBB	63.4	66.0	-	
	LFT	63.4	66.0	-	
	ONB	63.4	66.0	-	

Table 2: BERT NER-performance on different combinations of training and test sets. For all training/test pairs the same number of training epochs has been executed and the same learning parameters have been used.

Left (pre-train none): NER-performance of the non-modified multi-lingual BERT-Base model as provided by Google⁵.

Middle (pre-train DC-SBB): NER-performance of the multi-lingual BERT-Base model that has been pre-trained for 5 epochs on the DC-SBB data with objective “Masked-LM” and “Next Sentence Prediction” as proposed in (Devlin et al., 2018) prior to the NER supervised training.

Right (Riedl and Padó, 2018): NER-performance as published in (Riedl and Padó, 2018) where multiple state-of-the art CRF only and biLSTM + CRF approaches using different character and word embeddings have been evaluated.

Pre-training on DC-SBB improves results for historical German datasets, independently on the type of NER-ground-truth used for supervised training whereas the original BERT-base model provides better results on contemporary German test sets.

5-fold cross validation on	pre-train	BERT multi-lingual-cased			(Riedl and Padó, 2018)	(Schweter and Baiter, 2019)
		precision	recall	F_1	F_1	F_1
SBB	DC-SBB + GermEval + CoNLL	81.1 \pm 1.2	87.8 \pm 1.4	84.3 \pm 1.1	-	-
	DC-SBB + CoNLL	81.0 \pm 2.1	87.6 \pm 1.8	84.2 \pm 1.9	-	-
	DC-SBB + GermEval	80.6 \pm 1.8	87.4 \pm 1.3	83.8 \pm 1.2	-	-
	CoNLL	81.0 \pm 1.9	86.6 \pm 2.2	83.7 \pm 1.5	-	-
	GermEval	79.7 \pm 1.8	87.2 \pm 0.8	83.3 \pm 1.1	-	-
	GermEval + CoNLL	79.9 \pm 2.1	86.4 \pm 1.7	83.0 \pm 1.9	-	-
	DC-SBB	79.1 \pm 2.6	86.7 \pm 0.7	82.7 \pm 1.3	-	-
	none	79.1 \pm 3.6	85.0 \pm 1.1	81.9 \pm 2.2	-	-
	Newspaper (1703-1875)	-	-	-	-	85.31
ONB	DC-SBB+GermEval + CoNLL	81.5 \pm 1.8	87.8 \pm 1.4	84.6 \pm 1.5	-	-
	DC-SBB + GermEval	81.6 \pm 2.5	87.5 \pm 1.6	84.5 \pm 1.8	-	-
	DC-SBB + CoNLL	81.7 \pm 2.8	87.5 \pm 1.9	84.5 \pm 2.3	-	-
	DC-SBB	81.8 \pm 2.3	87.1 \pm 2.1	84.3 \pm 2.0	-	-
	GermEval	80.8 \pm 2.1	85.4 \pm 1.2	83.0 \pm 1.4	78.56	-
	GermEval + CoNLL	80.0 \pm 1.5	84.7 \pm 1.6	82.3 \pm 1.5	-	-
	CoNLL	79.1 \pm 2.5	84.5 \pm 2.1	81.7 \pm 2.2	76.17	-
	none	78.0 \pm 2.4	84.1 \pm 1.9	80.9 \pm 2.0	73.31	-
	Newspaper (1888-1945)	-	-	-	-	77.51
LFT	DC-SBB + CoNLL	70.0 \pm 2.6	81.0 \pm 0.7	75.1 \pm 1.5	-	-
	DC-SBB + GermEval	69.9 \pm 3.0	81.1 \pm 1.0	75.1 \pm 1.8	-	-
	DC-SBB	70.0 \pm 3.5	80.8 \pm 1.4	75.0 \pm 2.1	-	-
	DC-SBB + GermEval + CoNLL	69.8 \pm 3.0	80.8 \pm 0.9	74.9 \pm 2.0	-	-
	GermEval	68.9 \pm 2.7	79.3 \pm 1.4	73.7 \pm 1.9	74.33	-
	GermEval + CoNLL	69.1 \pm 2.6	78.8 \pm 1.3	73.6 \pm 1.5	-	-
	none	68.8 \pm 3.4	79.2 \pm 1.5	73.6 \pm 2.2	69.62	-
	CoNLL	68.4 \pm 3.1	79.1 \pm 1.3	73.3 \pm 2.1	72.9	-

Table 3: 5-fold cross validation results for different historical German NER corpora where different pre-training steps have been applied to the BERT model. For all experiments the same number of training epochs and the same learning parameters have been used. Results in (Riedl and Padó, 2018) and (Schweter and Baiter, 2019) have been obtained for some 80/20 training/test split.

None: Model as published by Google⁵.

DC-SBB: Model unsupervisedly pre-trained on DC-SBB.

CoNLL: Model supervisedly pre-trained on CoNLL training set.

GermEval: Model supervisedly pre-trained on GermEval training set.

DC-SBB + GermEval + CoNLL: First unsupervised pre-training for 5 epochs on the DC-SBB data. Second supervised pre-training on the joined GermEval and CoNLL NER ground truth.

The NER-performance under cross-validation can be significantly improved by combination of unsupervised and supervised pre-training. DC-SBB+GermEval+CoNLL pre-trained models show close to state-of-the-art performance on all three historical datasets using exactly the same training parameters and number of training epochs.

data leads to significantly different performances on varying test sets.

The original BERT model performs better than the biLSTM+CRF models in the case of contemporary training/test combinations. The pre-trained BERT model performs better than the biLSTM+CRF models in the case of the majority of historical training/test combinations except the CoNLL/ONB and GermEval/LFT pairs.

The impact of the diversity of the ground truth data sets makes it difficult to assess the actual performance of the BERT models on the historical data based on the results shown in Table 2 alone. In order to further study and clarify the experimental outcomes, another sequence of experiments was performed to evaluate the NER performance on the historical data under cross validation. The corresponding results are shown in Table 3. As above, the corresponding best results from (Riedl and Padó, 2018) are listed, if available, though their results have not been obtained under cross validation but for a fixed training/test split. (Schweter and Baiter, 2019) present a recent study of NER in historical German. They use a combined biLSTM + CRF model together with varying combinations of character embeddings, contextualized string embeddings (Akbik et al., 2018), pre-trained word embeddings, and BERT-layer features. We included their best results that have been obtained for a fixed train/test split on the LFT and ONB data set in the rightmost column of Table 3.

As illustrated by Table 3, various degrees of pre-training successively improve the performance of the BERT model. In case of the ONB and LFT data unsupervised pre-training alone (DC-SBB) provides the biggest part of improvement. Additional supervised pre-training adds only a small improvement. In case of the SBB ground truth, which is more similar to the contemporary data, supervised pre-training contributes more to the performance improvement.

BERT outperforms the biLSTM + CRF approaches that have been evaluated in (Riedl and Padó, 2018) but the results are still worse than some of the results reported in (Schweter and Baiter, 2019). Their best results rely on a pre-training scheme that is adapted to the final target domain whereas in our experiments the pre-training scheme DC-SBB + GermEval + CoNLL provides very good cross-validation performance for the three historical German sets SBB, ONB, and LFT while

utilizing the same set of learning parameters.

7 Conclusion and Future Work

The historical texts of the SBB digital collections originate from a broad period of time ranging from 1470 to 1945. A long term goal is to reliably conduct NER in this large text corpus in order to improve the user experience for researchers interacting with the library’s digitized holdings. Hence, a versatile approach is required that can deliver decent recognition performance for texts of different time epochs and a variety of text categories.

Our results show that an appropriately pre-trained BERT model delivers decent recognition performance in a variety of settings and even provides state of the art performance in many cases without extensive fine-tuning and optimization requirements. This outcome encourages further refinement and an extension of the methodology that has been evaluated in the presented experiments.

In the scope of this paper, we started all our experiments from the BERT-Base model. An increase of the model size is expected to improve the results further (Devlin et al., 2018). Therefore, we plan to re-run the experiments using BERT-Large which requires even more computation time.

In particular, the unsupervised pre-training on the DC-SBB set is computationally demanding. So far, we performed only 5 training epochs though further improvement in the unsupervised tasks “Masked-LM” and “Next Sentence Prediction” is still possible according to the trend of the loss. We plan to compensate for some of the additional computational demand by better and more GPU hardware that is currently installed at the SBB.

We think that there is a lot of performance to gain for historical text by adding more historical ground-truth data. Therefore, we plan to add more historical ground-truth data in the near future also in cooperation with the SoNAR project (Interfaces to Data for Historical Social Network Analysis and Research).

To end with, we plan to significantly reduce the level of noise in the source OCR texts by means of re-processing the digitized documents with LSTM OCR software specifically trained on historical texts and through the application of unsupervised OCR post-correction methods based on neural networks and finite-state-transducers being developed in the OCR-D project (Neudecker et al., 2019).

Acknowledgements

This work was partially supported by the German Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung, BMBF), project grant QURATOR - Curation Technologies⁶.

References

- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.
- Beatrice Alex and John Burns. 2014. Estimating and rating the quality of optically character recognised text. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, pages 97–102. ACM.
- Darina Benikova, Chris Biemann, Max Kisselew, and Sebastian Padó. 2014. Germeval 2014 named entity recognition: Companion paper. *Proceedings of the KONVENS GermEval Shared Task on Named Entity Recognition*, Hildesheim, Germany, pages 104–112.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT GitHub. <https://github.com/google-research/bert/blob/master/README.md>.
- Maud Ehrmann, Giovanni Colavizza, Yannick Rochat, and Frédéric Kaplan. 2016. Diachronic evaluation of ner systems on old newspapers. In *Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016)*, pages 97–107. Bochumer Linguistische Arbeitsberichte.
- Maria Federbusch, Christian Polzin, and Thomas Stäcker. 2013. Volltext via OCR- Möglichkeiten und grenzen. *Beiträge aus der Staatsbibliothek zu Berlin - Preußischer Kulturbesitz*, 43:1–138.
- Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL05)*, pages 363–370.
- Olivier Galibert, Sophie Rosset, Cyril Grouin, Pierre Zweigenbaum, and Ludovic Quintard. 2012. Extended named entities annotation on OCRed documents: from corpus constitution to evaluation campaign. In *International Conference on Language Resources and Evaluation*, Istanbul, Turkey, January.
- Claire Grover, Sharon Givon, Richard Tobin, and Julian Ball. 2008. Named entity recognition for digitised historical texts. *LREC 2008*.
- Hugging Face. 2019. BERT PyTorch GitHub. <https://github.com/huggingface/pytorch-pretrained-BERT>.
- Kimmo Kettunen and Teemu Ruokolainen. 2017. Names, right or wrong: Named entities in an ocred historical finnish newspaper collection. In *Proceedings of the 2nd International Conference on Digital Access to Textual Cultural Heritage*, pages 181–186. ACM.
- Kimmo Kettunen, Eetu Mäkelä, Teemu Ruokolainen, Juha Kuokkala, and Laura Löfberg. 2016. Old content and modern tools-searching named entities in a finnish ocred historical newspaper collection 1771–1910. *arXiv preprint arXiv:1611.02839*.
- Kai Labusch and David Zellhöfer. 2019. OCR Full-texts of the Digital Collections of the Berlin State Library (DC-SBB), June 26th. <https://doi.org/10.5281/zenodo.3257041>.
- Daniel Lopresti. 2009. Optical character recognition errors and their effects on natural language processing. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(3):141–151.
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- Marco Lui and Timothy Baldwin. 2012. Langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, ACL ’12, pages 25–30, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Clemens Neudecker, Lotte Wilms, Willem Jan Faber, and Theo van Veen. 2014. Large-scale refinement of digital historic newspapers with named entity recognition. In *Proceedings of the IFLA Newspapers/GENLOC Pre-Conference Satellite Meeting*.
- Clemens Neudecker, Konstantin Baierer, Volker Hartmann, Maria Federbusch, Matthias Boenig, and Elisa Hermann. 2019. OCR-D: An end-to-end open source ocr framework for historical printed documents. In *Proceedings of the Third International Conference on Digital Access to Textual Cultural Heritage*, page *in press*. ACM.
- Clemens Neudecker. 2016. An open corpus for named entity recognition in historic newspapers. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4348–4352, Portorož, Slovenia, May. European Language Resources Association (ELRA).
- Damien Nouvel, Jean-Yves Antoine, and Nathalie Friberger. 2011. Pattern mining for named entity recognition. In *Language and Technology Conference*, pages 226–237. Springer.

⁶<https://qurator.ai>

Thomas L Packer, Joshua F Lutes, Aaron P Stewart, David W Embley, Eric K Ringger, Kevin D Seppi, and Lee S Jensen. 2010. Extracting person names from diverse and noisy ocr text. In *Proceedings of the fourth workshop on Analytics for noisy unstructured text data*, pages 19–26. ACM.

Michael Piotrowski. 2012. Natural language processing for historical texts. *Synthesis lectures on human language technologies*, 5(2):1–157.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *arxiv*.

Martin Riedl and Sebastian Padó. 2018. A named entity recognition shootout for German. In *Proceedings of ACL*, pages 120–125, Melbourne, Australia.

Stefan Schweter and Johannes Baiter. 2019. Towards robust named entity recognition for historic german. *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP)*, page *in press*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL ’03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

A Supervised Learning Approach for the Extraction of Opinion Sources and Targets from German Text

Michael Wiegand^{*†}, Margarita Chikobava[†], Josef Ruppenhofer[‡]

^{*}Leibniz ScienceCampus, Heidelberg/Mannheim, Germany

[†]Spoken Language Systems, Saarland Informatics Campus,
Saarland University, Saarbrücken, Germany

[‡]Leibniz Institute for German Language, Mannheim, Germany

{wiegand|ruppenhofer}@ids-mannheim.de

margarita.chikobava@lsv.uni-saarland.de

Abstract

We present the first systematic supervised learning approach for the extraction of opinion sources and targets on German language data. A wide choice of different features is presented, particularly syntactic features and generalization features. We point out specific differences between opinion sources and targets. Moreover, we explain why implicit sources can be extracted even with fairly generic features. In order to ensure comparability our classifier is trained and tested on the dataset of the STEPS shared task.

1 Introduction

While there has been much research in sentiment analysis on typical text classification tasks, such as subjectivity detection, polarity classification and emotion classification, there has been notably less work on opinion role extraction. This particularly also concerns research done on languages other than English. In opinion role extraction, we distinguish between *opinion source extraction*, where the entities expressing an opinion are to be extracted, and *opinion target extraction*, where the task is to extract the entities or propositions at which sentiment is directed. For example, in (1) the sentiment expression *criticizes* has as its source *Switzerland* and as its target *North Korea*.

- (1) [Switzerland _{SOURCE}] **criticizes** [North Korea _{TARGET}].
- (2) [The opposition _{SOURCE}] **claims** [that the health service is getting fewer resources _{TARGET}].

In this work, we address opinion role extraction on German data. Research on this specific task and language has been kicked off by the *shared task on Source, Subjective Expression and Target Extraction from Political Speeches (STEPS)* with its two editions from 2014 (Ruppenhofer et al., 2014a) and 2016 (Ruppenhofer et al., 2016). Our experiments

are carried out on these data since, to the best of our knowledge, they are the only publicly available labeled data comprising annotation for opinion role extraction on German of sufficient size from which to train a classifier. These data also allow us to directly compare our work to systems that have participated in this shared task.

In this paper, we assume that the underlying sentiment expression which evokes opinion source or opinion target has already been identified. Decoupling role extraction from the identification of sentiment expressions seems reasonable to us since previous research has focused on subjectivity detection, i.e. the detection of sentiment expressions in context. The latter task is also considerably easier in which generic and resource-poor features yield good results. Even STEPS acknowledged this by offering a subtask where sentiment expressions are already provided and thus researchers may focus solely on opinion role extraction.

The **contributions** of this paper are that we present the first in-depth study to what extent different features are relevant for the task of opinion role extraction on German data. Since we present work on German language data this means that there exist fewer NLP tools and/or tools of lesser quality. We will examine which tools actually help. While most previous approaches only focused on the extraction of either sources or targets, we consider both entity types and highlight notable differences between these tasks. We also critically assess the amount of training data that is currently available. Finally, we conduct an evaluation against previous participations in the STEPS 2016 shared task to demonstrate the effectiveness of our approach.

We acknowledge that deep learning methods have recently received considerable attention in the NLP community. However, in this work we follow a more traditional feature-based approach employing supervised learning. The reason for this is that in the area of opinion role extraction, the

usage of deep learning methods has only produced moderate results (Katiyar and Cardie, 2016). A major caveat of deep learning methods is their reliance on distributional word representations (e.g. word embeddings). Opinion role extraction, however, is a task which relies on various types of linguistic information which are more expressive than the most robust word embeddings, such as syntactic dependency relations. Moreover, the amount of available training data for German is notably smaller than what is available for English (approximately by a factor of 10). This makes our setting fairly unfavourable for deep learning which usually outperforms traditional supervised approaches only if large amounts of labeled data are available.

2 Related Work

Like our proposed classifier, most previous approaches for opinion role extraction are supervised classifiers employing features from various information sources. They include surface-level information (Choi et al., 2005; Wiegand and Klakow, 2010), syntactic information (Choi et al., 2005; Kessler and Nicolov, 2009) and even information from semantic role labeling (Bethard et al., 2006; Kim and Hovy, 2006; Johansson and Moschitti, 2013). While particularly the latter type of information is very predictive for this task, we cannot apply it on our setting, since we are not aware of any robust semantic-role labeler for German.

Most previous research on opinion role extraction either only addressed opinion sources (Choi et al., 2005; Wiegand and Klakow, 2010; Johansson and Moschitti, 2013) or opinion targets (Kessler and Nicolov, 2009; Jakob and Gurevych, 2010). In this work, we look at both tasks. Thus we can show that there is a notable difference between these two tasks which also means that different classifier parameters and feature sets are required for those two different subtasks.

So far, work on opinion role extraction has mostly been carried out on English data. There has been some work on Chinese and Japanese as part of the NTCIR Opinion Analysis Task (Seki et al., 2007). Work on German that addresses both opinion source and target extraction has exclusively been carried out as part of the STEPS 2014 and 2016 shared tasks. There were few submissions made to the latter shared tasks. The systems presented can be divided into 3 different types:

- **rule-based approaches:** Wiegand et al.

(2014) present a system that works on extraction rules defined on sentiment expressions. The system applies heavy normalization of syntactic parse trees so that simple extraction rules cover a wide range of different sentences. Wiegand et al. (2016) is an extension of that system in which further components, such as a module to detect *grammatically induced sentiment*, are added. Despite only fairly generic extraction rules, this approach produced fairly good results.

- **translation-based approaches:** Wiegand et al. (2014) also present a second system which is a supervised learning system trained on the MPQA corpus which has been automatically translated into German. That approach notably suffers from the bad translation quality.
- **supervised approaches:** Both Kriese (2016) and Wiegand et al. (2016) present a supervised classifier. While Kriese (2016) proposes models that build on *path bundles* derived from a constituency parse tree, Wiegand et al. (2016) examine an SVM trained on various features including features from syntactic parses. The results are not very conclusive as no proper feature ablation studies are carried out.

Our work substantially extends previous supervised systems as we use more features (e.g. generalization features, features derived from a constituency parse tree, subcategorization features). Moreover, we optimize various parameters and features on some development set. Thus we ensure that the features and classifiers are used in their best possible configuration. Finally, we conduct various experiments examining different feature subsets. These experiments are vital in order to make general conclusions regarding which type of information is really required for this task.

3 Data & Annotation

For our experiments we employ the labeled datasets from the STEPS 2014 shared task (Ruppenhofer et al., 2014b) and the STEPS 2016 shared task (Ruppenhofer et al., 2016) comprising 605 and 580 sentences, respectively. For STEPS 2016, the STEPS 2014 dataset was revised in order to be compatible with the new annotation scheme introduced for STEPS 2016. We use this revision of the STEPS 2014 dataset. The advantage of using datasets from the revised annotation scheme is that this scheme

Property	Freq
number of sentences	1185
average length of sentence	21
sentiment expressions	4646
sentiment expr. with neither source nor target	753
number of sources	3402
number of targets	3378
proportion of development set	10%

Table 1: Statistics of the dataset.

has been shown to produce a sufficiently high inter-annotation agreement (Ruppenhofer et al., 2016).

Since both datasets are fairly small, we merged them and conduct our experiments on the union of both datasets. Table 1 provides some descriptive statistics of our resulting dataset. 10% of the dataset were reserved as development data. On this data, we optimized various features and parameters of our classifier (§7.1).

4 Classifier and Instance Space

We pursue a supervised learning approach and decided in favor of using SVMs. As a tool, we employ SVM^{light} (Joachims, 1999). We consider the extraction of sources and targets as two completely separate tasks.

Both sources and targets always relate to a specific sentiment expression which evokes them. Therefore our instance space comprises tuples of sentiment expression and candidate opinion source phrase for sources, and sentiment expression and candidate target phrase for targets (Table 2). As a candidate source phrase, we consider all noun phrases (NPs) and preposition phrases (PPs) from the sentence in which the given sentiment expression occurs, while for targets, we consider any constituent of a sentence to be a candidate. This difference can be explained by the fact that only persons qualify as a source (hence NPs and PPs) while targets represent a more heterogeneous class of entities. For example, in (1) it is an NP representing a country while in (2) it is a complement clause representing a proposition.

5 Implicit Opinion Sources

A considerable number of opinion sources in our dataset are implicit. That is, there is no constituent in the relevant sentence that represents this opinion source. Instead the opinion source is the speaker of the utterance. For example, in (3) the sentiment expression *offensichtlich* (*obvious*) has no explicit source.

- (3) [Die Gründe dafür _{TARGET}] sind **offensichtlich**.
([The reasons for that _{TARGET}] are **obvious**.)

The likelihood of an opinion source being implicit very much depends on its sentiment expression. For example, a word such as *obvious* will predominantly have an implicit source. Table 3 shows the distributions of the different source types according the part of speech of their sentiment expressions. There is clearly a correspondence. For example, of all parts of speech the likelihood of implicit sources is highest with adjectives.¹ A classifier that takes into account the part of speech of sentiment expressions is already able to make good guesses as to the presence or not of an explicit source (for example by predicting all opinion adjectives as having an implicit source and all opinion nouns having an explicit source). Further, the lexical knowledge of sentiment expressions as a feature will also be beneficial. For example, we found that more than one third of the verbal sentiment expressions having implicit sources are evoked by verbs conveying so-called *grammatically-induced sentiment* (Wiegand et al., 2016). This concerns sentiment that is conveyed by certain modalities (4)-(5).

- (4) [Deshalb **müssen** wir diesen Prozess stärker ankurbeln.
_{TARGET}]
([That is why we **must** to crank this process up. _{TARGET}])
(5) [Sie **sollten** hier ein Signal setzen. _{TARGET}]
([You **should** send a clear message here. _{TARGET}])

Such sentiment is evoked by frequently occurring auxiliary and modal verbs, such as *werden* (*will*) or *sollen* (*should*). Even on comparatively small training corpora, such as ours, this information can be directly learned. That is, no manual lexicon is required for detecting such cases of sentiment as the precision of those verbs to predict an implicit source on our dataset is about 94%.

In order to enable our supervised learner to predict implicit sources, we simply need to adjust the instance space for opinion sources. In addition to explicit constituents from the sentence (see discussion above), we also add a dummy instance with an empty candidate source phrase. These instances will represent implicit sources. Indeed our exploratory experiments on the development set, as shown in Table 4, confirmed that just adding dummy instances for sources with our full feature

¹We found that the actual proportion of implicit sources on that part of speech is actually even higher, since many sentiment adjectives having an explicit source actually turned out to be verbs erroneously tagged as adjectives.

Role	Instance	Candidate Phrases
source	<sentiment expr., candidate phrase>	all NPs, PPs and an empty dummy phrase for implicit sources (§5)
target	<sentiment expr., candidate phrase>	all phrases of a sentence

Table 2: Instances for opinion sources and targets (*the sentiment expression is always given*).

Source	Adj		Noun		Verb	
	Freq	Perc	Freq	Perc	Freq	Perc
explicit	154	27.2	1411	80.1	1164	71.1
implicit	412	72.8	350	19.9	472	28.9

Table 3: Parts of speech of implicit sources.

w/o Implicit Instances			w Implicit Instances		
Prec	Rec	F1	Prec	Rec	F1
56.6	27.6	37.1	55.8	49.7	52.6

Table 4: Impact of implicit sources instances.

set (that includes the above features describing the part of speech and the lemma of the sentiment expression) drastically increases extraction performance for source extraction.

6 Feature Design

Our feature set is too large for us to be able to perform an evaluation on each individual feature. Instead, we group our features according to **5 meaningful dimensions** and evaluate them. In the following, we discuss those dimensions. Our complete feature set is heavily based on features employed for opinion role extraction in English. For more motivation of our feature set, we therefore refer the reader to previous work, particularly by Choi et al. (2005) and by Kessler and Nicolov (2009).

The first dimension groups our features according to the linguistic representation on which they are based. For instance, there are features that encode some semantic information, others describe syntactic or just surface-based information.

The second dimension is the focus of the feature. We distinguish between features that describe the *individual* linguistic entities involved in role extraction, that is, the sentiment expression or the candidate source/target phrase; features that describe their *relation*; and features that describe further *context* (i.e. features that focus on words other than the sentiment expression or candidate phrase).

Our third dimension divides the feature set into *simple* and *complex* features. By complex features, we understand features that require the usage of some lexical resource or a computationally intensive NLP tool (here we consider every tool more

complex than a part-of-speech tagger).

The fourth dimension states whether a feature *generalizes* some lexical information or not. The generalization may be produced in a data-driven way (e.g. Brown clustering (Brown et al., 1992)) or with the help of some lexical resource (e.g. GermaNet (Hamp and Feldweg, 1997)).

The final dimension groups our feature set according to the information source it uses. By information source, we define the resource or NLP tool that is used in order to extract a particular feature. Table 5 lists all features we use and also characterizes them according to each dimension.

For part-of-speech tagging we used *TreeTagger* (Schmid, 1994), for constituency parsing the *Berkeley Parser* (Petrov et al., 2006), for dependency parsing *ParZu* (Sennrich et al., 2009), for named-entity recognition, we used the tagger by Faruqui and Padó (2010). The Brown clusters were induced with the help of *SRILM* (Stolcke, 2002). We induced 1000 clusters from the *HGC corpus*². As a subcategorization lexicon, we used *IMSLex* (Fitschen, 2004).

7 Experiments

7.1 Parameter Optimization

Before we examine the different feature subsets, we need to optimize some feature and classification parameters. For these experiments, we always test a classifier on the development data. The classifiers are trained on the remaining data. We now list these optimized settings:

- Best level of generalization for GermaNet hypernyms (we do not just consider the direct hypernyms but also higher-up ancestors): for both sources and targets we consider hypernyms up to their third ancestors.
- Best cut-off value for length of part-of-speech sequences: 5 for sources; all sequences for targets.
- Best cut-off value for length of constituency paths: 5 for sources; 10 for targets.
- Best cut-off value for length of dependency relation paths: 5 for sources; 5 for targets.
- Best cost-parameter that adjusts the classifier to the imbalanced class distribution: $j=5$ for sources; $j=6$ for targets. (In opinion role extraction, like all entity extraction tasks, the entities to be extracted represent a

²<http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/hgc.html>

Feature	Representation	Focus	Dimensions		
			Simplicity	Generalizing	Info. Source
head of sentiment expr. head of candidate phrase context as bag of words	word word word	individual individual context	simple simple simple	no no no	lexical unit lexical unit lexical unit
Is candidate phrase first phrase of sentence? orientation of candidate phrase in relation to sentiment expr. distance between candidate phrase and sentiment expr.	surface	individual	simple	no	other
	surface	relation	simple	no	other
	surface	relation	simple	no	other
cluster id of head of sentiment expr. cluster id of head of candidate phrase cluster ids of context words	semantic	individual	complex	yes	Brown clustering
	semantic	individual	complex	yes	Brown clustering
	semantic	relation	complex	yes	Brown clustering
named entity of candidate phrase	semantic	individual	complex	yes	named-entity tagging
synset id(s) of head of sentiment expr. synset id(s) of head of candidate phrase GermaNet word class of head of sentiment expr. GermaNet word class of head of candidate phrase GermaNet word class of words in context	semantic	individual	complex	yes	GermaNet
	semantic	individual	complex	yes	GermaNet
	semantic	individual	complex	yes	GermaNet
	semantic	individual	complex	yes	GermaNet
	semantic	relation	complex	yes	GermaNet
pos of head of sentiment expr. pos of head of candidate phrase pos sequence between candidate phrase and sentiment expr.	syntactic syntactic syntactic	individual individual relation	simple simple simple	no no no	pos tagging pos tagging pos tagging
subcategorization frame according to subcat. lexicon number of arguments on subcategorization frame according to subcat. lexicon	syntactic	individual	complex	no	subcat. lexicon
	syntactic	individual	complex	no	subcat. lexicon
phrase label of candidate phrase tuple of phrase label of candidate phrase and pos of head of sentiment expr. pos-tuple of head of candidate phrase and head of sentiment expr. subcategorization frame derived from constituency tree number of arguments in subcategorization frame derived from constituency tree constituency label path between heads of candidate phrase and sentiment expr. length of constituency label path between heads of candidate phrase and sentiment expr.	syntactic syntactic	individual relation	complex complex	no no	constituency parsing constituency parsing
	syntactic	relation	simple	no	constituency parsing
	syntactic	individual	complex	no	constituency parsing
	syntactic	individual	complex	no	constituency parsing
	syntactic	relation	complex	no	constituency parsing
	syntactic	relation	complex	no	constituency parsing
subcategorization frame derived from dependency tree number of arguments on subcategorization frame derived from dependency tree dependency relation path between heads of candidate phrase and sentiment expr. length of dependency relation path between head of candidate phrase and sentiment expr.	syntactic	individual	complex	no	dependency parsing
	syntactic	individual	complex	no	dependency parsing
	syntactic	relation	complex	no	dependency parsing
	syntactic	relation	complex	no	dependency parsing

Table 5: Features and their categorization along 5 dimensions.

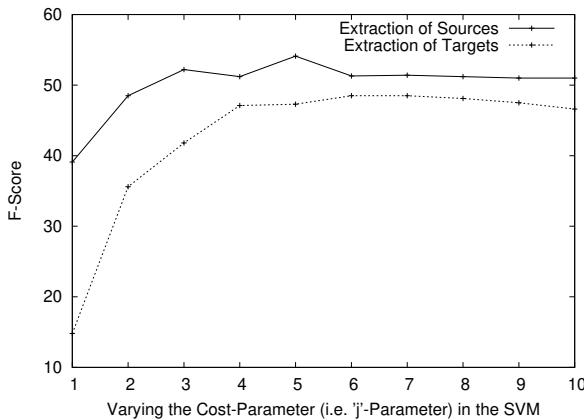


Figure 1: Optimizing the cost parameter.

minority class. This typically results in datasets with very imbalanced class distributions.)

We exemplify the importance of this optimization on the cost-parameter. Figure 1 shows the different F-scores of different cost-parameters for both source and target extraction on the development set. Clearly, the default value (i.e. $j = 1$) would only produce poor results of the classifier.

7.2 Comparison of Different Feature Groups

Given the optimal configurations determined in §7.1, we now examine the different feature groups on a 10-fold crossvalidation. We report macro-average precision, recall and F(1)-score.

Table 6 shows the performance of the individual foci and their combinations. This analysis shows that the most important focus is the set of relational features. Adding other features only yields mild increases in performance. The table also shows that regarding the other foci, there is a notable difference between the tasks. While for extraction of sources, both *individual* and *context* provide some decent F-score, on the extraction of targets they are not useful at all. While it is difficult to explain this behaviour for the context features, we found some intuitive explanation for the behaviour of the *individual* features. Opinion sources are per definition a very restricted set of entities sharing specific semantic properties. That is, only persons or groups of persons qualify as opinion sources. Therefore, a personal pronoun or the mention of a proper name (notice that our *individual* features capture this type of information), will already have a relatively high prior probability of representing a source. Targets, on the other hand, represent a

Subset	Source			Target		
	Prec	Rec	F1	Prec	Rec	F1
individual	48.2	41.2	44.4	5.4	0.2	0.3
relational	59.5	49.8	54.2	47.5	40.1	43.5
context	44.7	33.1	38.0	38.2	4.1	7.4
ind.+rel.	59.4	51.6	55.2	48.8	40.1	44.0
ind.+cont.	48.3	44.7	46.4	31.6	12.7	18.0
rel.+cont.	56.4	47.4	51.5	47.8	38.8	42.9
<i>all</i>	56.0	54.0	55.0	49.1	39.9	44.0

Table 6: Comparison of different foci.

Subset	Source			Target		
	Prec	Rec	F1	Prec	Rec	F1
<i>all</i>	56.0	54.0	55.0	49.1	39.9	44.0
-clustering	55.7	52.4	54.0	49.0	39.6	43.8
-GermaNet	57.0	51.6	54.1	49.9	40.6	44.7
-depend.	55.4	53.0	54.2	45.3	37.3	41.2
-constit.	55.1	49.7	52.3	46.1	35.4	40.0
-subcat	56.3	53.7	55.0	49.1	39.8	44.0
-pos	55.8	52.6	54.2	49.5	38.5	43.3
-named ent.	56.0	53.6	54.8	49.2	40.0	44.1
-other	56.1	52.3	54.1	50.6	39.1	44.1
-lexical	60.1	51.4	55.7	49.1	40.3	44.3
-dep.-const.	52.2	47.6	49.8	39.6	32.6	35.8

Table 7: Ablation experiments.

much more heterogeneous group. They may be entities of various semantic types, they may even be represented by propositions (cf. (1) and (2)). This explains why targets are more dependent on relational features. That is, they can be more easily identified by their relationship towards the existing sentiment expression. For example, in both (1) and (2), the target is an object of its sentiment expression.

Table 7 shows some ablation experiments in which we remove one information source at a time. This gives us an indication of how unique the individual information sources are in terms of the information they contribute to the prediction of sources and targets. Only few information sources seem to carry unique information. The most notable exceptions are dependency and constituency parse information. On target extraction, we notice a notable drop in performance if either of those types of features are removed. We also removed both of these information sources at the same time to show that dependency and constituency information are not only important but are also complementary to each other.

Table 8 compares the performance of the different linguistic representations. The results are in line with the previous experiments. Word-level features are much more predictive for sources than for targets. Virtually all those features are *individual* features, so the explanation that we provided in Table 6 also applies here. Although word-level,

Subset	Source	Target
word	42.7	6.9
word+semantic	45.3	17.1
word+surface	47.2	24.4
word+semantic+surface	48.0	26.4
word+syntactic	53.6	44.2
word+surface+syntactic	53.6	44.1
<i>all</i>	55.0	44.0

Table 8: F-Scores of different linguistic representations.

Subset	Source			Target		
	Prec	Rec	F1	Prec	Rec	F1
simple	53.6	46.4	49.8	40.7	34.7	37.5
complex	59.9	49.7	54.3	50.3	29.9	37.5
<i>all</i>	56.0	54.0	55.0	49.1	39.9	44.0

Table 9: Simple and complex features.

semantic and surface features can be effectively combined, the most notable boost in performance is obtained when syntactic features are added. This is in line with our ablation experiments (Table 7) where we found that constituency and dependency parsing, in other words, *syntactic* features carry the most distinct information for this task.

Table 9 compares simple and complex features. Again, we observe notable differences between source and target extraction. While the two feature groups are on a par on target extraction, on source extraction the complex features are stronger. The combination of the feature groups is more effective on target extraction than on source extraction.

Table 10 shows the impact of generalization of both tasks. There is no clear indication that the generalization features actually help. Particularly on the extraction of targets these features are not useful at all. We explain the latter results by the fact that the generalizations are basically generalizations of the *individual* features and we pointed out in the discussion of Table 6 that those features seem to not be predictive for targets. A generalization of a completely unrelated feature is very likely to be not predictive either.

7.3 Learning Curve

The amount of labeled training data that is available to us (i.e. about 1,200 sentences) is still very small.

Subset	Source			Target		
	Prec	Rec	F1	Prec	Rec	F1
plain	57.5	51.2	54.2	49.5	40.1	44.3
generalizat.	47.1	39.3	42.9	4.1	0.2	0.3
<i>all</i>	56.0	54.0	55.0	49.1	39.9	44.0

Table 10: The impact of generalization.

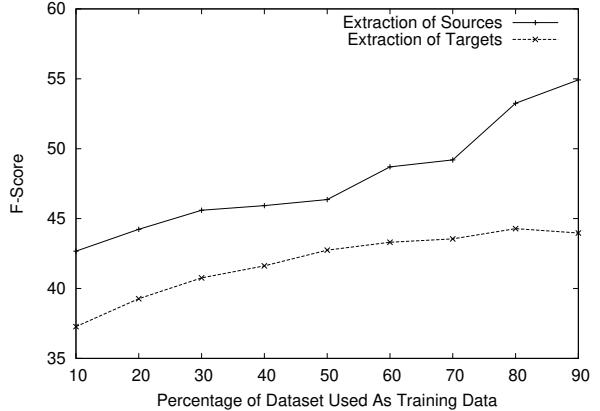


Figure 2: Learning curve on gold standard.

Because of this, we computed a learning curve in order to estimate in how far increasing the amount of labeled training data would affect classification performance. Figure 2 displays a learning curve. While for sources, the curve clearly indicates that a larger amount of labeled training data is likely to increase classification performance, for targets the curve seems to be almost saturated. We already argued above that the extraction of targets is considerably more difficult than the extraction of sources. Presumably, source extraction would benefit from more labeled training data since then the classifier could get more evidence of which nouns or noun phrases are likely opinion sources and which are not. We strongly assume that due to the semantic heterogeneity of targets, such features are not effective no matter what amount of training data is available. With regard to relational/syntactic features, the current amount of labeled training data may be sufficient since there are only a handful of meaningful syntactic relationships holding between a sentiment expression and either of its sources or targets (e.g. *subject*, *object* etc.).

7.4 Comparison against Previous Classifiers

Finally, we evaluate our classifier with the full feature set against other systems that participated in the STEPS 2016 shared task. In order to produce a meaningful comparison, unlike our previous experiments, we train our classifier only on the training data from that shared task.³ Table 11 shows the performance of the different classifiers. Overall, our proposed supervised system produces the best per-

³This explains why the performance of our proposed system is slightly lower than in the previous experiments.

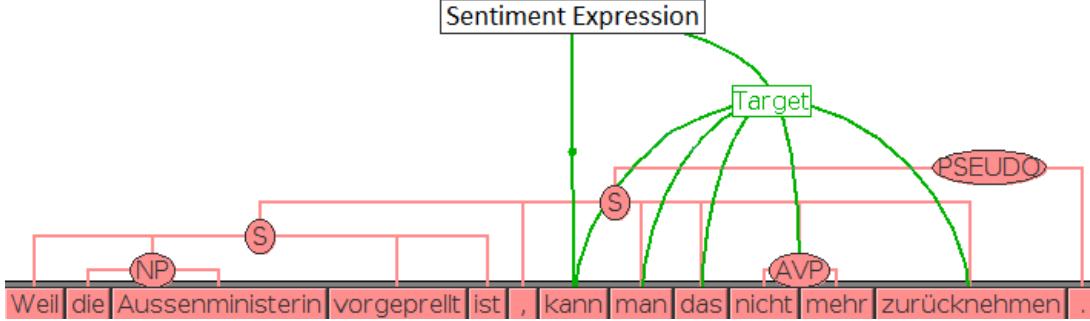


Figure 3: Illustration of opinion target covering *more* than one constituent.

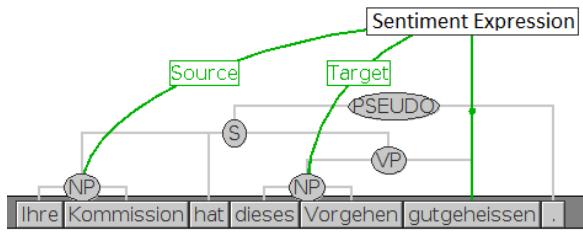


Figure 4: Illustration of opinion target covering *exactly* one constituent.

formance. While on the extraction of sources, we notably outperform all other classifiers, on the extraction of targets, the rule-based system from Saarland University (Wiegand et al., 2016) is on the par with our classifier. This classifier is able to recognize instances of opinion targets that our system is unable to recognize. It concerns cases of so-called *grammatically induced sentiment* (§5). In such cases, the target typically is an entire (sub)clause. In the output of a constituency parser, these clauses often correspond to more than one constituent as illustrated in Figure 3. However, our classifier always assumes one constituent per source and target each as illustrated in Figure 4. Therefore, our approach is unable to correctly extract the above targets. In future work, we would like to combine that classifier with ours in order to hopefully obtain even a higher classification performance.

7.5 Error Analysis

Unfortunately, it is outside the scope of this paper to provide an in-depth error analysis. However, we could identify the output of syntactic parsing as a major source of error. We established in our evaluation that syntactic features are most predictive. Given that completely correct syntactic analyses on our data are rare it comes as no surprise that

System	Prec	Source Rec	F1
Saarland University (supervised)	59.4	38.3	46.6
Saarland University (rule based)	59.9	28.6	38.7
Potsdam University (supervised)	36.2	30.0	32.9
<i>proposed system</i>	58.0	44.0	50.3
System	Prec	Target Rec	F1
Saarland University (supervised)	42.6	31.7	36.3
Saarland University (rule based)	69.2	28.9	40.8
Potsdam University (supervised)	37.3	22.2	27.8
<i>proposed system</i>	48.1	35.0	40.5

Table 11: Comparison with systems of the STEPS 2016 Shared Task.

the overall classification performance we achieve is still comparatively low.

8 Conclusion

We presented a supervised learning approach for opinion role extraction for German. We found that there are notable differences between the extraction of opinion sources and opinion targets. Opinion targets are more difficult to handle. Even with comparably simple features, opinion sources can be extracted. For both tasks, information describing the relation between the given sentiment expression and the candidate opinion role, particularly the information drawn from syntactic parses, is most important. Generalization features do not increase classification performance much. Even though our feature set is not specifically tailored to implicit opinion sources, we are able to detect a considerable proportion. Our best classifier outperforms the best classifier which participated in the STEPS 2016 shared task. With regard to opinion target extraction, it performs on a par with the best previously reported classifier.

Acknowledgements

This research has been partially supported by the Leibniz ScienceCampus “Empirical Linguistics and Computational Modeling”, funded by the Leibniz Association under grant no. SAS-2015-IDS-LWC and by the Ministry of Science, Research, and Art (MWK) of the state of Baden-Württemberg. The authors were also partially supported by the German Research Foundation (DFG) under grants RU 1873/2-1 and WI 4204/2-1.

References

- Steven Bethard, Hong Yu, Ashley Thornton, Vasileios Hatzivassiloglou, and Dan Jurafsky. 2006. Extracting Opinion Propositions and Opinion Holders using Syntactic and Lexical Cues. In *Computing Attitude and Affect in Text: Theory and Applications*, pages 125–141. Springer-Verlag.
- Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.
- Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying Sources of Opinions with Conditional Random Fields and Extraction Patterns. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 355–362, Vancouver, BC, Canada.
- Manaal Faruqui and Sebastian Padó. 2010. Training and Evaluating a German Named Entity Recognizer with Semantic Generalization. In *Proceedings of KONVENS*, Saarbrücken, Germany.
- Arne Fitschen. 2004. *Ein computerlinguistisches Lexikon als komplexes System*. Ph.D. thesis, Institut für maschinelle Sprachverarbeitung, Universität Stuttgart.
- Birgit Hamp and Helmut Feldweg. 1997. GermaNet – a Lexical-Semantic Net for German. In *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15, Madrid, Spain.
- Niklas Jakob and Iryna Gurevych. 2010. Extracting Opinion Targets in a Single- and Cross-Domain Setting with Conditional Random Fields. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1035–1045, Boston, MA, USA.
- Thorsten Joachims. 1999. Making Large-Scale SVM Learning Practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press.
- Richard Johansson and Alessandro Moschitti. 2013. Relational Features in Fine-Grained Opinion Analysis. *Computational Linguistics*, 39(3):473–509.
- Arzoo Katiyar and Claire Cardie. 2016. Investigating LSTMs for Joint Extraction of Opinion Entities and Relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 919–929, Berlin, Germany.
- Jason S. Kessler and Nicolas Nicolov. 2009. Targeting Sentiment Expressions through Supervised Ranking of Linguistic Configurations. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*, San Jose, CA, USA.
- Soo-Min Kim and Eduard Hovy. 2006. Extracting Opinions, Opinion Holders, and Topics Expressed in Online News Media Text. In *Proceedings of the ACL Workshop on Sentiment and Subjectivity in Text*, pages 1–8, Sydney, Australia.
- Leonard Kriese. 2016. System documentation for the IGGSA Shared Task 2016. In *Proceedings of IGGSA Shared Task Workshop*, volume Bochumer Linguistische Arbeitsberichte, pages 10–13, Bochum, Germany.
- Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the International Conference on Computational Linguistics and Annual Meeting of the Association for Computational Linguistics (COLING/ACL)*, pages 433–440, Sydney, Australia.
- Josef Ruppenhofer, Roman Klinger, Julia Maria Struß, Jonathan Sonntag, and Michael Wiegand. 2014a. IGGSA Shared Tasks on German Sentiment Analysis (GESTALT). In G. Faaß and J. Ruppenhofer, editors, *Workshop Proceedings of the KÖNVENS Conference*, pages 164–173, Hildesheim, Germany. Universität Hildesheim.
- Josef Ruppenhofer, Julia Maria Struß, Jonathan Sonntag, and Stefan Gindl. 2014b. IGGSA-STEPS: Shared Task on Source and Target Extraction from Political Speeches. *Journal for Language Technology and Computational Linguistics*, 29(1):33–46.
- Josef Ruppenhofer, Julia Maria Struß, and Michael Wiegand. 2016. Overview of the IGGSA 2016 Shared Task on Source and Target Extraction from Political Speeches. In *Proceedings of IGGSA Shared Task Workshop*, Bochumer Linguistische Arbeitsberichte, pages 1–9, Bochum, Germany.
- Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49, Manchester, United Kingdom.
- Yohei Seki, David Kirk Evans, Lun-Wei Ku, Hsin-Hsi Chen, Noriko Kando, and Chin-Yew Lin. 2007.

Overview of Opinion Analysis Pilot Task at NTCIR-6. In *Proceedings of the NTCIR-6 Workshop Meeting*, Tokyo, Japan.

Rico Sennrich, Gerold Schneider, Martin Volk, and Martin Warin. 2009. A New Hybrid Dependency Parser for German. In *Proceedings of the German Society for Computational Linguistics and Language Technology (GSCL)*, pages 115–124, Potsdam, Germany.

Andreas Stolcke. 2002. SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, pages 901–904, Denver, CO, USA.

Michael Wiegand and Dietrich Klakow. 2010. Convolution Kernels for Opinion Holder Extraction. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*, pages 795–803, Los Angeles, CA, USA.

Michael Wiegand, Christine Bocioneck, Andreas Conrad, Julia Dembowski, Jörn Giesen, Gregor Linn, and Lennart Schmeling. 2014. Saarland University’s Participation in the GERman SenTiment AnaLysis shared Task (GESTALT). In G. Faaß and J. Ruppenhofer, editors, *Workshop Proceedings of the KONVENS Conference*, pages 174–184, Hildesheim, Germany. Universität Hildesheim.

Michael Wiegand, Nadisha-Marie Aliman, Tatjana Anikina, Patrick Carroll, Margarita Chikobava, Erik Hahn, Marina Haid, Katja König, Leonie Lapp, Artuur Leeuwenberg, Martin Wolf, and Maximilian Wolf. 2016. Saarland University’s Participation in the Second Shared Task on Source, Subjective Expression and Target Extraction from Political Speeches (STEPS-2016). In *Proceedings of IGGSA Shared Task Workshop*, Bochumer Linguistische Arbeitsberichte, pages 14–23, Bochum, Germany.

A Descriptive Analysis of a German Corpus Annotated with Opinion Sources and Targets

Michael Wiegand^{*†}, Leonie Lapp[†], Josef Ruppenhofer[‡]

^{*}Leibniz ScienceCampus, Heidelberg/Mannheim, Germany

[†]Spoken Language Systems, Saarland Informatics Campus,
Saarland University, Saarbrücken, Germany

[‡]Leibniz Institute for German Language, Mannheim, Germany

{wiegand|ruppenhofer}@ids-mannheim.de

leonie.lapp@lsv.uni-saarland.de

Abstract

We present a descriptive analysis on the two datasets from the shared task on Source, Subjective Expression and Target Extraction from Political Speeches (STEPS), the only existing German dataset for opinion role extraction of its size. Our analysis discusses the individual properties of the three components, subjective expressions, sources and targets and their relations towards each other. Our observations should help practitioners and researchers when building a system to extract opinion roles from German data.

1 Introduction

While there has been much research in sentiment analysis on typical text classification tasks, such as subjectivity detection, polarity classification and emotion classification, there has been notably less work on opinion role extraction. This particularly concerns research done on languages other than English. In opinion role extraction, we distinguish between *opinion source extraction*, where the entities expressing an opinion, i.e. the opinion sources, are to be extracted, and *opinion target extraction*, where the task is to extract the entities or propositions at which sentiment is directed, i.e. the opinion targets. For example, in (1) the subjective expression *criticizes* has as its source *Switzerland* and as its target *North Korea*.

- (1) [Switzerland SOURCE] **criticizes** [North Korea TARGET].
- (2) [The opposition SOURCE] **claims** [that the health service is getting fewer resources TARGET].

In this paper, we address opinion role extraction on German data. Research on this specific task and language has been kicked off by the *shared task on Source, Subjective Expression and Target Extraction from Political Speeches (STEPS)* with its two editions from 2014 (Ruppenhofer et al., 2014a;

and 2016 (Ruppenhofer et al., 2016). We present a descriptive analysis of the two datasets from this shared task that serve as a gold standard for opinion role extraction on German. Our aim is not to produce a classifier to automatically extract opinion sources and targets. Instead, we look into the properties of this gold standard in order to guide researchers and practitioners who intend to build such a classifier. Our analysis should largely influence the choice of classifiers, particularly the underlying feature set that describes potential opinion roles.

The focus of our analysis is on the structure of the opinion frame (§3), i.e. the linguistic structure that relates opinion source and target to its subjective expression. For each of these three linguistic components (subjective expression, opinion source and opinion target), we look at their individual linguistic forms and also their (syntactic) relation towards each other. Since STEPS consists of two datasets, i.e. the editions from 2014 and 2016, we also compare in how far the observed properties between the two different datasets differ. Given that each dataset individually is very small (i.e. only about 600 annotated sentences) the combination of the two datasets is a desirable step when building a classifier for opinion role extraction. Only if the two datasets are compatible to a large degree, can they be used for building a single application.

For general accessibility, we will always provide English examples when German and English follow the same linguistic pattern. Since, to the best of our knowledge, this is the first descriptive corpus-based study for opinion role extraction in general, we believe that our insights may be relevant to research beyond the German language.

Syntactic information plays a significant role in opinion role extraction, particularly, dependency relations. In this work, we consider dependency parses produced by ParZu (Sennrich et al., 2009). We consider this parser since it is also the dependency parser which the organizers of STEPS em-

ploy in the release of their dataset.

2 Related Work

So far, work on opinion role extraction has mostly been carried out on English data, especially the MPQA corpus (Wiebe et al., 2005), the standard corpus for fine-grained sentiment analysis. There has also been a related shared task on the topic: the Sentiment Slot Filling track (SSF) that was part of the Shared Task for Knowledge Base Population of the Text Analysis Conference (TAC) (Mitchell, 2013). For Japanese and Chinese some comparable data have been created as part of the NTCIR Opinion Analysis Task (Seki et al., 2007; Seki et al., 2008; Seki et al., 2010).

To the best of our knowledge, the only descriptive analysis of opinion role extraction was presented by Ruppenhofer et al. (2008). The major difference to our work is that Ruppenhofer et al. (2008) enumerate linguistic phenomena involved in opinion role extraction without reference to some existing datasets. Since we examine a labeled corpus, our main contribution is that we quantify the linguistic phenomena involved.

For German sentiment analysis, there exist quite a few different corpora ranging from sentiment aspect classification (Säger et al., 2016; Wojatzki et al., 2017) to much more fine-grained tasks, such as attitude classification (Klenner et al., 2017). Apart from STEPS, however, there only exists the MLSA corpus (Clematide et al., 2012) with annotation of both opinion holders and targets on German text. The annotation scheme of STEPS was mainly inspired by *Layer 3: Expression-level Annotation* of MLSA. (The same researchers who annotated that layer of MLSA also created the two STEPS datasets.) The reason we conduct our study on the STEPS corpus rather than on the MLSA corpus is that the two STEPS corpora totaling about 1,200 sentences are significantly larger than the MLSA corpus with only 270 sentences.

3 Opinion Frames

In STEPS, opinion roles are represented as *opinion frames*. An opinion frame is a triple $\langle \text{subjective expression}, \text{opinion source}, \text{opinion target} \rangle$. The *subjective expression* is a word or phrase which conveys some opinion, its *source* is the entity that expresses that opinion, and its *target* is the entity or proposition towards which that opinion is directed. In this paper, subjective expressions will always

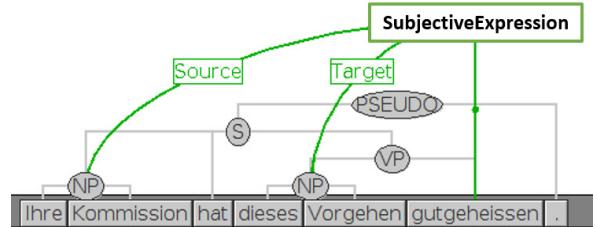


Figure 1: Illustration of an opinion frame.

be indicated by bold type font in examples and abbreviated by the acronym SE in the prose.

Each opinion frame has exactly one SE and at most one source and target each. In other words, there can be opinion frames without a source (3), without a target (4) or without both (5).

- (3) I don't understand this **interest** [in weapons **TARGET**].
- (4) [Peter **SOURCE**] was so **unhappy** that he immediately left the party.
- (5) **Altruism** is not a very common thing in our society.

4 IGGSA-STEPS: Shared Task on Source and Target Extraction from Political Speeches

For our experiments we employ the labeled datasets from the *Shared Task on Source and Target Extraction from Political Speeches*. In that shared task, German language speeches from the Swiss parliament were annotated with opinion frames. In German, there exists no comparable dataset of similar size for opinion role extraction. The data have been annotated in TIGER/SALSA XML (Erk and Padó, 2004), a format originally devised for representing frame-structures from FrameNet (Baker et al., 1998). This representation format combines syntactic constituency parses with some semantic annotation. Like FrameNet-frames, opinion frames represent semantic structures that build upon syntactic structures. Figure 1 illustrates the structure of a typical opinion frame.

There are two editions of the shared task (Ruppenhofer et al., 2014b; Ruppenhofer et al., 2016). For STEPS 2016, the STEPS 2014 dataset was revised in order to be compatible with the new annotation scheme introduced for STEPS 2016. We use this revision of the STEPS 2014 dataset. Another advantage of using the dataset from the revised annotation scheme is that it has been shown to produce a sufficiently high interannotation agreement (Ruppenhofer et al., 2016).

Table 1 displays some general statistics of the two datasets. The table already indicates that there

property	freq	
	STEPS 2014	STEPS 2016
no. of sentences	605	581
avg. no. of tokens in sentence	22.58	24.08
no. of subjective exprs. (SEs)	2105	2166
avg. no. of SEs in sentence	3.58	3.94
no. of opinion frames	2228	2417
no. of sources	997	1064
no. of targets	1608	1770

Table 1: Statistics of the two STEPS datasets.

are no significant differences in the frequency of the different major constructions between STEPS 2014 and STEPS 2016.

5 Subjective Expressions (SEs)

The first step in opinion role extraction is to determine which words represent SEs. Table 2 provides some statistics about this linguistic entity. The most notable observation is that many SEs are singletons. (This ratio does not change much even if we merge the two datasets STEPS 2014 and STEPS 2016.) This is highly relevant for building a classifier to detect SEs. If most SEs only occur once in a gold standard, then they can hardly be learnt from this data directly. Instead, some form of sentiment lexicon listing SEs should be used. However, by computing the coverage of the SEs in the standard sentiment lexicon for German, the PolArt lexicon (Klenner et al., 2009), we found that only a small proportion (i.e. 25%) is actually covered.

With about 19% of the vocabulary, multiword expressions (MWEs) represent a considerable share in the set of SEs. About 75% are MWEs that consist of exactly 2 tokens, which, in most cases, are phrasal verbs, e.g. *tritt ab* (*stands down*), *denkt nach* (*considers*). Compared to idioms, e.g. *in den sauren Apfel beißen* (*to bite the bullet*), which due to their free word order in German can have many different surface realizations (Wiegand et al., 2016a), phrasal verbs are relatively easy to detect.

Table 3 lists the distribution of the different parts of speech among the SEs. To our surprise, nouns are the most frequent type of SEs. One typically associates sentiment with adjectives (e.g. *bad*, *nice*) or verbs (e.g. *adore*, *hate*) and, therefore, one would expect a higher proportion of these parts of speech. The high frequency of subjective nouns can be explained by the fact that many of these nouns are nominalized adjectives (e.g. *badness*) and nominalized verbs (e.g. *hatred*). Additionally, many subjective nouns are some form of compound, e.g. *Bombenattentat* (*bombing attack*) or *Expertenmei-*

dataset	types	singletons	MWEs
STEPS 2014	1115	805	213
STEPS 2016	1110	769	214

Table 2: Statistics of subjective expressions (SEs).

dataset	verb	adj	noun	adv	other
STEPS 2014	270	206	418	18	227
STEPS 2016	280	207	405	27	224

Table 3: POS-Distribution of SEs (*types are counted*).

nung (*expert advice*). Wiegand et al. (2016b) state that every other sentence in STEPS 2014 contains a noun compound.¹ A noun compound (*Bombenattentat*) typically consists of two constituents, a modifier (*Bombe*) and a head (*Attentat*). Noun compounds are very productive. In principle, noun heads may combine with a large number of different noun modifiers (e.g. *Bombenattentat*, *Selbstmordattentat*, *Flugzeugattentat*, *Sprengstoffattentat*, *Säureattentat* etc.) This results in a large number of different compounds in STEPS (please keep in mind that in Table 3, we count types and not tokens). Each of these compounds only occurs once or twice on average which explains the high number of noun types, particularly singleton nouns in STEPS.

In order to detect SEs automatically and given the large number of sparse noun compounds on both datasets, some form of noun normalization would be advisable. Only the head of a noun compound is relevant for detecting SEs, i.e. *Attentat* (*attack*) in *Bombenattentat* (*bombing attack*). We, therefore, anticipate a higher coverage of matching SEs in a sentiment lexicon by reducing noun compounds to their heads.

If one pursues a lexicon-based approach to detect SEs, not only is a lexicon sought that has a good coverage, one should also keep the reliability of the entries in mind. Table 4 compares the precision of an oracle lexicon, i.e. a lexicon comprising all words being labeled at least once as an SE expression in either of the two editions of STEPS, with the precision of the words in the PolArt sentiment lexicon. (We evaluate here on the concatenation of STEPS 2014 and STEPS 2016. In the following sections, we always merge the distributions of STEPS 2014 and STEPS 2016 whenever there was not sufficient space and we did not observe any significant difference between the two datasets.)

¹We also confirm a similar proportion in STEPS 2016.

lexicon	Prec
union of words being labeled as SE at least once	72.1
PolArt lexicon	89.9

Table 4: Precision of different lexicons.

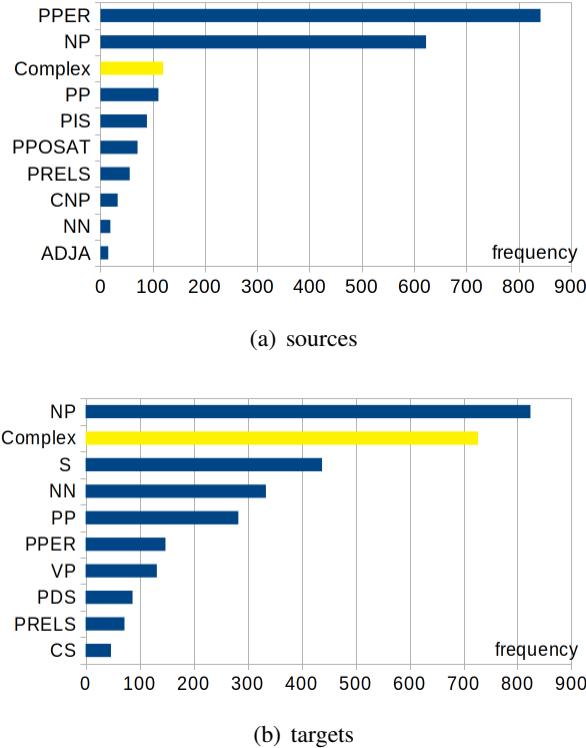


Figure 2: Distribution of phrase labels.

Although the PolArt sentiment lexicon has a low coverage of SEs in STEPS, the entries that match that lexicon are fairly reliable. A lexicon with a full coverage of SEs (as our oracle lexicon) would not solve the problem of detecting SEs, as a large proportion of SEs are ambiguous words. One additionally would have to devise a subjectivity word-sense disambiguation (Akkaya et al., 2009) once a word within a sentence has been matched with that lexicon. The task would be to decide whether an ambiguous word, such as *alarm* is used in a subjective sense, as in (6), or in a non-subjective one, as in (7). If no reliable disambiguation is possible, a sentiment lexicon may still be a good solution because of its high precision.

- (6) When he heard that particular news, his **alarm** grew even more.
- (7) Our new smoke detector is malfunctioning. The **alarm** went off twice yesterday although there was no smoke.

6 Inherent Properties of Opinion Roles

We now examine properties of opinion sources and targets. We start by looking at inherent properties. Figure 2 compares the phrase label distribution of opinion sources and targets. Typically, both source and target exactly match one phrase node in the constituency parse tree representing the sentence (Figure 1). We introduce a phrase label *Complex* by which we subsume all cases in which an opinion role does not match exactly one constituent. A large fraction of those instances will be parse errors.²

Figure 2 shows that sources and targets have notably different phrase labels. Opinion sources are mostly noun phrases (NP) or personal pronouns. This result is quite intuitive. Opinion sources can only be persons or groups of persons as other types of entities typically do not have any specific sentiment. The fact that prepositional phrases are also frequent can be explained by passive constructions (9) in which the opinion source is realized as a prepositional phrase rather than an NP which is the case in the more canonical active constructions (8).

(8) [Peter _{SOURCE}] **loves** [Mary _{TARGET}]^{NP}.

(9) [Mary _{TARGET}] is **loved** [by Peter _{SOURCE}]^{PP}.

Opinion targets, on the other hand, are much more heterogeneous. Targets do not have to be entities. They can also represent entire propositions. This explains why other constituents, such as (complement) sentences (10) or verbal phrases (11), are also frequently labeled as targets.

(10) [Peter _{SOURCE}] **thinks** [that Mary should work harder _{TARGET}]^S.

(11) [Peter _{SOURCE}] **wants** [to go shopping _{TARGET}]^{VP}.

It is also surprising that the second most frequent phrase label is *Complex*. By manually inspecting these cases, we found that in most of them there was an error in the parse. Targets representing entire propositions are typically much longer phrases (i.e. they comprise more tokens) than source-phrases representing simple entities. Figure 3 illustrates the different token lengths of sources and targets. It confirms that sources tend to be shorter than targets. The long phrases that represent targets, such as sentences or verbal phrases,

²The constituency-parse trees in STEPS have been automatically generated by the Berkeley parser (Petrov and Klein, 2007). In case of parsing errors, the annotators were instructed to label those spans of text that they thought represent the correct span. This often meant that one opinion role was assigned more than one phrase node in the constituency-parse tree.

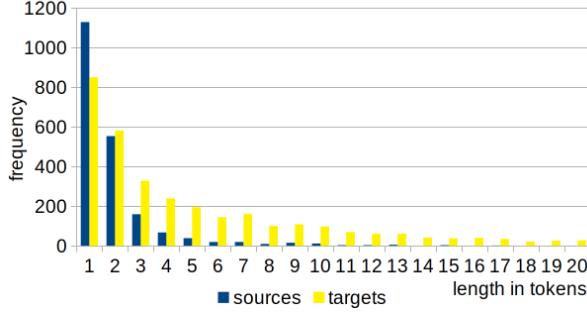


Figure 3: Distribution of phrase length.

are also much more likely to be affected by parsing errors.

7 Opinion Roles and Their Relation towards SEs

We continue our examination of opinion roles by looking at the relation between opinion roles to the SEs they evoke. We start by looking into syntactic relationships.

Table 5 lists the 5 most frequent dependency-relation paths observed between the individual opinion roles and the SEs they evoke. The table lists the paths for subjective verbs, adjectives and nouns each. It suggests that some dependency-relation paths are predictive for either sources or targets. For example, for subjective verbs, sources are often realized as subjects ($\uparrow subj$) while targets are realized as accusative objects ($\uparrow obja$), as illustrated by (12). For subjective attributive adjectives, one can very reliably predict targets by looking for the noun that modifies them ($\downarrow attr$) as illustrated by (13).

- (12) [Mary SOURCE]^{subj} **likes**_{verb} [Peter’s new flat TARGET]^{obj}.
 (13) I just saw a **beautiful**_{adj}^{attr} [rainbow TARGET].

However, there are also relation paths that are ambiguous. The most notable example is the genitive modifier of subjective nouns ($\uparrow gmod$) which is the most frequent dependency relationship connecting both opinion sources (14) and targets (15). This analysis proves that opinion roles cannot be extracted exclusively on the basis of dependency-relation paths.

- (14) Das entsprach auch der **Sichtweise**_{noun} [der meisten Bürger SOURCE]^{gmod}.
(This was also the view of most citizens.)
 (15) Er hob die **Einfachheit**_{noun} [des Ansatzes TARGET]^{gmod} hervor.
(He emphasized the simplicity of that approach.)

pos	relation path	ratio [%]		
		sources	targets	freq
verb	$\uparrow subj$	81.2	18.8	479
	$\downarrow aux_up\uparrow subj$	67.9	32.1	324
	$\uparrow obja$	7.7	92.3	221
	$\uparrow pp$	12.6	87.4	87
adj	$\uparrow objd$	21.2	78.8	52
	$\downarrow attr$	1.3	98.7	235
	$\downarrow pred_up\uparrow subj$	30.9	69.1	55
	$\downarrow aux_up\uparrow subj$	69.2	30.8	26
	$\downarrow adv_up\uparrow subj$	33.3	66.7	21
noun	$\uparrow pp$	66.7	33.3	12
	$\uparrow gmod$	38.4	61.6	199
	$\uparrow pp$	15.2	84.8	79
	$\downarrow obja\uparrow subj$	68.9	31.1	74
	$\uparrow det$	76.9	23.1	65
	$\uparrow attr$	48.0	52.0	25

Table 5: Ambiguity of dependency-relation paths between sources and targets.

One major obstacle in processing German text is the high degree of errors in automatic syntactic parse analyses. The longer a sentence is the more likely errors in syntactic parsing occur. This certainly is an issue with STEPS 2014 and STEPS 2016 since both datasets contain long sentences (between 22 and 24 tokens on average, see also Table 1). The ParZu parser may fail to produce a fully connected dependency tree for long sentences. Instead only a set of partial trees are produced. In that case, there is often no dependency-relation path available that connects opinion roles with the SEs they evoke.

For all pairs of $<opinion\ role, SE>$ where the members of the pair are separated by a specified token distance, Figure 4 shows the proportion of pairs for which there is no connecting dependency path available in the output of ParZu. The figure shows that the longer the token distance is the higher the proportion of pairs are that have no dependency-relation path. Even for pairs with a short token distance, there is still a considerable number of pairs for which there is no dependency-relation path. All in all, this analysis underlines that errors in the syntactic parse output will have an impact on classification performance.

As an alternative to syntactic dependency-relation paths, we also examine whether the order of pairs $<opinion\ role, SE>$ is predictive for this task. Unlike syntactic information, information about the sequential order of two constituents is not dependent on the output of a syntactic parser. Table 6 displays the ratio of different orders. The table shows that there may be certain correlations between certain orders. For example, the source mostly precedes subjective verbs or adjec-

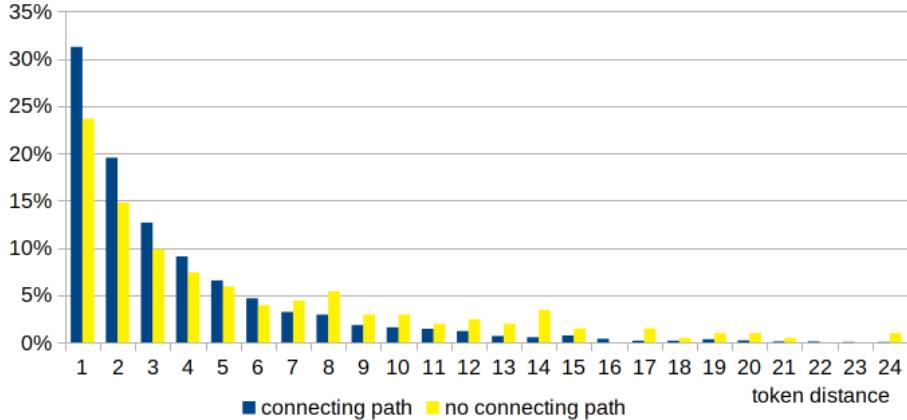


Figure 4: Connecting paths between opinion role and SE for different token lengths.

order	STEPS 2014			STEPS 2016		
	verb	adj	noun	verb	adj	noun
<source, SE>	81.7	91.1	62.0	83.4	97.6	55.9
<SE, source>	18.3	8.9	38.0	16.6	2.4	44.1
<target, SE>	67.5	24.2	32.1	66.8	36.5	30.9
<SE, target>	32.5	75.8	67.9	33.2	63.5	69.1

Table 6: Order of opinion role (i.e. source or target) and SE (in percentage).

tives. However, in the case of targets, these correlations are less pronounced. The general lack of a predictive sequential order can be explained by the fact that depending on tense or sentence type, the order between different constituents may vary. For instance, the canonical order for subjective verbs *<SE, opinion target>* as can be observed in a present tense main clause (16) changes if that sentence is shifted into present perfect (17) or a subordinate clause (18).

- (16) Peter **hasst**_{verb} [Julia TARGET]^{obja}.
(Peter hates Julia.)
- (17) Er hat schon immer [Julia TARGET]^{obja} **gehasst**_{verb}.
(Peter has always hated Julia.)
- (18) ... weil Peter [Julia TARGET]^{obja} **hasst**_{verb}.
(... because Peter hates Julia.)

However, in all of these cases, the dependency relation between subjective expression and opinion target remains the same ($\uparrow obja$). This example illustrates that, in principle, syntactic dependency relations are more expressive than sequential order.

8 Frame Structure Configurations

According to the definition of opinion frames (§3), the presence of both source and target is not obligatory. We want to examine how often the opinion frame structure deviates from the canonical form

in which both source and target are present. Table 7 lists the frequency of different frame structure configurations. The table clearly shows that both in STEPS 2014 and STEPS 2016 there is a significant number of frames that do not include both source and target. This observation is quite important since it suggests that joint modelling of opinion source and target using simple constraints of the type *an opinion frame has to comprise exactly one opinion source and one opinion target* would not work.

In Table 7, we also observe that partial frames with only a target are much more frequent than the frames with only a source. We ascribe this large amount to the so-called *implicit* sources. Implicit sources are sources without a concrete surface realization (19). They typically represent the speaker of the utterance in which the opinion frame is evoked. Strictly speaking, therefore, frames with such a source are not partial frames. These frames just lack an *explicit* source, that is, a constituent in the sentence in which the SE occurs which has been annotated as an opinion source. Whether an SE comes with an explicit or implicit source largely depends on the SE itself. In other words, it is a lexical property of SEs. For English, Wiegand et al. (2016c) developed methods to distinguish whether an SE is more likely to have explicit or implicit sources. While SEs with a tendency for implicit sources are called *speaker-views* SEs, SEs with a tendency for explicit sources are referred to as *actor-views* SEs. Most of these methods should be largely reproducible on German language data.

Apart from opinion frames with implicit sources, there may, of course, also be opinion frames lack-

	STEPS dataset	
frame structure	2014	2016
frames with source and target	845	850
frames with only source	152	214
frames with only target	763	920
frames with neither source nor target	468	433

Table 7: Distribution of different frame structures.

ing both an explicit and implicit source. An example of the latter type is (20). It does not contain an explicit source and from the context it is clear that it is not the speaker of the utterance either since the speaker (represented by *I*) explicitly distances themselves from the interest in weapons. Therefore, the exact source remains unspecified.

- (19) [The reasons for voting to leave the EU TARGET] are **obvious**.
- (20) I don't understand this **interest** [in weapons TARGET].

Figure 5 compares the distribution of frames without source and frames without target across SEs with different parts of speech. While for verbs, we observe fewer frames that exclude either source or target, we observe that for nouns and adjectives partial frame structures are much more frequent. (This also matches our previous examples (3)-(5).) Particularly, most frames without a target are evoked by subjective nouns. The fact that mostly adjectives and nouns are likely to form partial opinion frames might be explained with the help of subcategorization. Although the subcategorization frames of verbs and nouns can be similarly complex (for instance, both the subjective verb in (21) and the subjective noun in (22) have two arguments), for verbs the realization of its arguments is usually obligatory in order to make a sentence grammatical (cp. (21) with (23)). For nouns (and adjectives follow similarly), however, it is quite often the case that they come with fewer arguments than their valency suggests (24). The fewer arguments a subjective expression has, the more likely partial frames are to be evoked. (24) contains a partial frame lacking a target.

- (21) [Mary SOURCE]^{subj} **loves**_{verb} [Peter TARGET]^{obj}.
- (22) Everyone knew about [Mary's SOURCE]^{gmod} **love**_{noun} [to Peter TARGET]^{pobj}.
- (23) ?[Mary SOURCE]^{subj} **loves**_{verb}.
- (24) In public, only few people talk about [Mary's SOURCE]^{gmod} **love**_{noun}.

9 Inferred Sources

Most opinion roles are syntactic dependents of the SE by which they are evoked. For instance, the

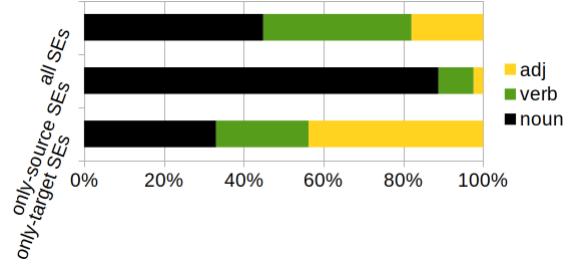


Figure 5: POS-Distribution of partial frames.

source of *like* in (25) is its subject. In STEPS, there is a special subset of sources, referred to as *inferred sources*. By that we understand sources that are not associated with any of the syntactic dependents of its SE (26). (In (26), the SE *impressive* has only one syntactic dependent which is its subject.) These sources are called *inferred* since from the subcategorization frame of the SE, we cannot conclude their presence. This makes them more difficult to detect than normal sources.

- (25) [Mary SOURCE]^{subj} **likes**_{verb} [Peter's new flat TARGET]^{obj}.
- (26) [Mary ^{INFERRED_SOURCE}] said [Peter's new flat TARGET]^{subj} was **impressive**_{adj}.

26% of the opinion sources in STEPS 2014 and STEPS 2016 have been flagged as inferred sources by the annotators. Since this is a substantial amount, we want to investigate whether we can further characterize this subset of sources. If we look at the distribution of parts of speech of the SEs evoking inferred sources (Figure 6), we find that there is a notable difference to the general part-of-speech distribution of SEs. While the proportion of nouns remains fairly constant, there is a disproportionately high amount of inferred sources with subjective adjectives. For SEs being verbs, the proportion of inferred sources, on the other hand, is fairly low.

We assume that the valency of the individual SEs is responsible for that distribution. The prototypical (subjective) adjective has one syntactic argument, for example, a subject (27) which is its target. There is no argument position for the opinion source and therefore, the source is the implicit speaker of the utterance.³ However, if this SE is

³The source in this sentence is not unspecified, since *impressive* in (27) comes with all its obligatory syntactic arguments, i.e. its subject. According to Wiegand et al. (2016c) more than 90% of all subjective adjectives are *speaker-view* words, i.e. these are subjective expressions that tend to have implicit sources.

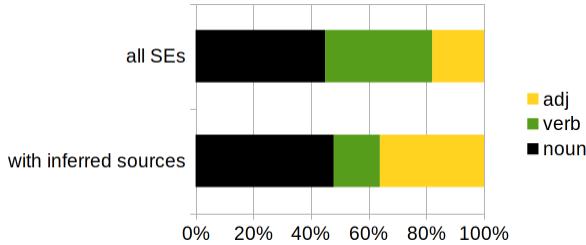


Figure 6: POS-Distribution of SEs with inferred sources.

further syntactically embedded, as in (26), there may be some explicit source but it is inferred. For subjective verbs, unlike subjective adjectives, there is a syntactic argument in their subcategorization frame, typically their subject, as in (25), that is associated with their opinion source. Therefore, fewer inferred sources occur with subjective verbs.

- (27) [Peter’s new flat _{TARGET}]^{subj} is **impressive**_{adj}.

10 Multiple Frame Evocation

There are SEs that evoke more than one opinion source and target. In STEPS this is modeled by allowing the same SE to evoke more than one single opinion frame. For example, the verb *force* can evoke three different opinion frames at the same time as illustrated by (28)-(30). (28) describes the view that James has some request to someone. (29) describes the view of James towards walking the dog. Finally, (30) represents Alice’s negative sentiment towards walking the dog (if she did not have that sentiment, James would not need to force her to do so).

- (28) [James _{SOURCE}] **forced** [Alice _{TARGET}] to walk the dog.
 (29) [James _{SOURCE}] **forced** Alice [to walk the dog _{TARGET}].
 (30) James **forced** [Alice _{SOURCE}] [to walk the dog _{TARGET}].

12% of the SEs in STEPS evoke more than one opinion frame. Figure 7 shows the distribution of multiple frame evocation across SEs with different parts of speech. The statistic shows that by far most SEs evoking multiple frames are verbs. This can be explained by the fact that verbs have the most complex subcategorization frames (e.g. in (28)-(30) *force* has three different syntactic arguments). We assume that the more syntactic arguments a SE has in a sentence, the more likely there is some multiple frame evocation.

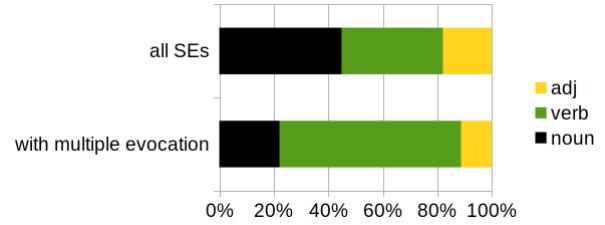


Figure 7: POS-Distribution of SEs with multiple frame evocation.

11 Conclusion

We presented a descriptive analysis of the STEPS 2014 and 2016 datasets, a resource for building and evaluating opinion role extraction systems in German. We found that the linguistic properties of the two datasets are very similar which means that they can be usefully merged into one resource. A large proportion of subjective expressions are nouns including noun compounds. The majority of subjective expressions are singletons. We assume that in order to increase the coverage of subjective expressions in lexical resources, such as sentiment lexicons, more effectively, some noun normalization that reduces compounds to their heads may be helpful. Opinion sources and targets differ very much from each other. Opinion sources tend to be realized as (short) noun phrases, while opinion targets are long phrases of various types. For both opinion sources and targets there is a small set of characteristic dependency relationships towards the subjective expression they evoke. Conceptually speaking, dependency relationships are more predictive than sequential order. However, reliable syntactic information is difficult to produce since parsers for German are fairly error prone. STEPS includes a substantial number of inferred sources. Those subjective expressions that come with inferred sources have more often few syntactic arguments, such as adjectives. Subcategorization frames also play a role when it comes to partial opinion frames. Subjective expressions with very complex subcategorization frames, such as verbs, typically come with complete opinion frames unlike adjectives and nouns, which more often evoke partial opinion frames. There is also a significant number of subjective expressions that evoke multiple frames, however, this phenomenon is largely restricted to subjective verbs.

Acknowledgements

This research has been partially supported by the Leibniz ScienceCampus “Empirical Linguistics and Computational Modeling”, funded by the Leibniz Association under grant no. SAS-2015-IDS-LWC and by the Ministry of Science, Research, and Art (MWK) of the state of Baden-Württemberg. The authors were also partially supported by the German Research Foundation (DFG) under grants RU 1873/2-1 and WI 4204/2-1.

References

- Cem Akkaya, Janyce Wiebe, and Rada Mihalcea. 2009. Subjectivity Word Sense Disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 190–199, Singapore.
- Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the International Conference on Computational Linguistics and Annual Meeting of the Association for Computational Linguistics (COLING/ACL)*, pages 86–90, Montréal, Quebec, Canada.
- Simon Clematide, Stefan Gindl, Manfred Klenner, Stefanos Petrakis, Robert Remus, Josef Ruppenhofer, Ulli Waltinger, and Michael Wiegand. 2012. MLSA – A Multi-layered Reference Corpus for German Sentiment Analysis. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, pages 3551–3556, Istanbul, Turkey.
- Katrin Erk and Sebastian Padó. 2004. A powerful and versatile xml format for representing role-semantic annotation. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, pages 799–802, Lisbon, Portugal.
- Manfred Klenner, Angela Fahrni, and Stefanos Petrakis. 2009. PolArt: A Robust Tool for Sentiment Analysis. In *Proceedings of the Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 235–238, Odense, Denmark.
- Manfred Klenner, Simon Clematide, and Don Tuggener. 2017. Verb-Mediated Composition of Attitude Relations Comprising Reader and Writer Perspective. In *Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, Lecture Notes in Computer Science, pages 141–155, Budapest, Hungary. Springer.
- Margaret Mitchell. 2013. Overview of the TAC2013 Knowledge Base Population Evaluation: English Sentiment Slot Filling. In *Proceedings of the Text Analysis Conference (TAC)*, Gaithersburg, MD, USA.
- Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*, pages 404–411, Rochester, NY, USA.
- Josef Ruppenhofer, Swapna Somasundaran, and Janyce Wiebe. 2008. Finding the Source and Targets of Subjective Expressions. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, pages 2781–2788, Marrakech, Morocco.
- Josef Ruppenhofer, Roman Klinger, Julia Maria Struß, Jonathan Sonntag, and Michael Wiegand. 2014a. IGGSA Shared Tasks on German Sentiment Analysis (GESTALT). In G. Faaß and J. Ruppenhofer, editors, *Workshop Proceedings of the KONVENTS Conference*, pages 164–173, Hildesheim, Germany. Universität Hildesheim.
- Josef Ruppenhofer, Julia Maria Struß, Jonathan Sonntag, and Stefan Gindl. 2014b. IGGSA-STEPS: Shared Task on Source and Target Extraction from Political Speeches. *Journal for Language Technology and Computational Linguistics*, 29(1):33–46.
- Josef Ruppenhofer, Julia Maria Struß, and Michael Wiegand. 2016. Overview of the IGGSA 2016 Shared Task on Source and Target Extraction from Political Speeches. In *Proceedings of IGGSA Shared Task Workshop*, Bochumer Linguistische Arbeitsberichte, pages 1–9, Bochum, Germany.
- Mario Säger, Ulf Leser, Steffen Kemmerer, Peter Adolphs, and Roman Klinger. 2016. SCARE - The Sentiment Corpus of App Reviews with Fine-grained Annotations in German. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, pages 1114–1121, Portorož, Slovenia.
- Yohei Seki, David Kirk Evans, Lun-Wei Ku, Hsin-Hsi Chen, Noriko Kando, and Chin-Yew Lin. 2007. Overview of Opinion Analysis Pilot Task at NTCIR-6. In *Proceedings of the NTCIR-6 Workshop Meeting*, Tokyo, Japan.
- Yohei Seki, David Kirk Evans, Lun-Wei Ku, Hsin-Hsi Chen, and Noriko Kando. 2008. Overview of Multilingual Opinion Analysis Task at NTCIR-7. In *Proceedings of the NTCIR-7 Workshop Meeting*, Tokyo, Japan.
- Yohei Seki, Lun-Wei Ku, Le Sun, Hsin-Hsi Chen, , and Noriko Kando. 2010. Overview of Multilingual Opinion Analysis Task at NTCIR-8 - A Step Toward Cross Lingual Opinion Analysis. In *Proceedings of NTCIR-8 Workshop Meeting*, pages 209–220.
- Rico Sennrich, Gerold Schneider, Martin Volk, and Martin Warin. 2009. A New Hybrid Dependency Parser for German. In *Proceedings of the German Society for Computational Linguistics and Language Technology (GSCL)*, pages 115–124, Potsdam, Germany.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating Expressions of Opinions and Emotions in Language. *Language Resources and Evaluation*, 39(2/3):164–210.

Michael Wiegand, Nadisha-Marie Aliman, Tatjana Anikina, Patrick Carroll, Margarita Chikobava, Erik Hahn, Marina Haid, Katja König, Leonie Lapp, Artuur Leeuwenberg, Martin Wolf, and Maximilian Wolf. 2016a. Saarland University’s Participation in the Second Shared Task on Source, Subjective Expression and Target Extraction from Political Speeches (STEPS-2016). In *Proceedings of IGGSA Shared Task Workshop*, Bochumer Linguistische Arbeitsberichte, pages 14–23, Bochum, Germany.

Michael Wiegand, Christine Bocionek, and Josef Ruppenhofer. 2016b. Opinion Holder and Target Extraction on Opinion Compounds – A Linguistic Approach. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*, pages 800–810, San Diego, CA, USA.

Michael Wiegand, Marc Schulder, and Josef Ruppenhofer. 2016c. Separating Actor-View from Speaker-View Opinion Expressions using Linguistic Features. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*, pages 778–788, San Diego, CA, USA.

Michael Wojatzki, Eugen Ruppert, Sarah Holschneider, Torsten Zesch, and Chris Biemann. 2017. GermEval 2017: Shared Task on Aspect-based Sentiment in Social Media Customer Feedback. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 1–12, Berlin, Germany.

Automated Assessment of Language Proficiency on German Data

Edit Szügyi and Sören Etler and Andrew Beaton and Manfred Stede

Cognitive Systems Program
Applied Computational Linguistics
University of Potsdam / Germany

`szuegyi|etler|beaton|stede@uni-potsdam.de`

Abstract

The proficiency level of the learner is an important factor in various educational settings. In order to find the adequate language difficulty level, we classify texts written by language learners of German into proficiency levels A, B and C, as defined by the CEFR (Common European Framework of Reference for Languages), based on linguistic features extracted from the texts. Working on a combined data set of previously-used corpora, we use both data- and theory-driven feature sets, and determine the best-performing features. Our model achieves an accuracy of 82%, and the best-performing feature set contains features from all the theoretical groups, while all groups alone perform significantly above the random baseline.

1 Introduction

An important concept in the field of educational systems is Automatic Text Scoring (ATS), which automates the process of scoring texts by using NLP techniques. A special case of ATS is Automatic Proficiency Assessment (APA), which aims at scoring texts written by language learners according to a proficiency scale; in Europe, this is defined by the Common European Framework of Reference for Languages (CEFR). With the help of APA, educators can more easily find appropriate reading materials and students can get immediate feedback on their performance. Furthermore, perhaps we can also get closer to a more practical definition of the CEFR levels by way of linguistic feature extraction.

In the scope of this project, we have developed an APA system that classifies diverse German texts written by language learners into levels A, B and C of the CEFR. Level A (elementary), includes

CEFR levels A1 and A2, Level B (intermediate), consists of levels B1 and B2 and level C (advanced) is composed of levels C1 and C2. We implement a wide range of linguistic features, which are described in Section 3.

2 Related Work

The earliest large-scale APA systems for German have been developed in the work of Hancke (2013) (see also Hancke, Vajjala and Meurers (2012)). She implements lexical, morphological, syntactic and language model features, building on work from different languages as well as different but highly related fields, such as Second Language Acquisition and Readability Assessment. Her feature sets are theoretically well-motivated and exhaustive. One aspect of her work that we think can be improved concerns the size and imbalance of the data set, the MERLIN (Wisniewski et al., 2011). While we also include it in our study, we overcome some of the problems by using a larger and balanced data set. Hancke achieved 72.5% accuracy working on 5 classes, A1–C1, and our overall goal is to build on and expand her research with new analyses.

As for other authors who work on German readability, Vajjala (2013) tests readability features on German text books in her PhD thesis, using the readability features developed by Hancke et al. (2012). Lavalley and Kay (2014) use children’s writing as their data and work with embellishment clues (adjectives and adverbs) as features. Nietzio et al. (2012) work with texts written for mentally challenged people, and use sentence length and complexity features. Brück and Hartrumpf (2007) work with legal texts and semantic features. Zesch et al. (2015) use English and German texts to test which features are independent of the specific writing tasks or prompts. One very recent piece of work is by Weiss and Meurers (2018), who use media texts for children and adults with the goal

of implementing a linguistically broad readability model for German. Their features are from the fields of lexicon, syntax, morphology, discourse, language use and human processing.

Among studies of texts written by learners of other languages, the majority – as can be expected – has focused on English. Yannakouadis (2011) works on grading texts written by learners of English with lexical features, part of speech (POS) tags, syntax features, length features and error rate. Treffers et al. (2018) study the correlation between lexical diversity and CEFR levels. Briscoe et al. (2010) analyze machine learning methods better suited for the task, using n-grams, parse rules, word length and error measures. It is also important to consider the possible end users of this line of research, namely educators, students, and readers. With that in mind, Chen and Meurers (2016) provide a publicly available platform for automatic complexity feature extraction and visualization.

3 Methods

3.1 Overview

In our work, we have implemented a wide range of features based on Hancke’s thesis (2013), but using a bigger data set (see Section 4). While one of our goals is to develop a classification system, what we think is even more important is a thorough discussion of the performance of the feature space, and see how it relates to Hancke (2013), which is the only other piece of work discussing similar features in a similar setting.

We work with texts written by learners of German and implement a supervised classification model according to the CEFR categories A, B and C as labels. In order to train the model, we have experimented with different machine learning algorithms, such as Decision Trees, Logistic Regression and Support Vector Machines (SVM) with different configurations. We have decided to use a Linear SVM as it performed best given our data set. This was an expected outcome, as SVMs perform well in various classification settings, both inside and outside NLP. Other researchers in the field of automatic readability and proficiency assessment also found them to give the best results (Hancke, 2013; Pilán et al., 2016; Zesch et al., 2015; Weiss and Meurers, 2018)

We have used the implementation of scikit-learn (Pedregosa et al., 2011) with its default settings. Since SVMs are sensitive to the distribution of

the data, we have balanced our data set. We end up working with 612 texts for each level, so our data set consists of 1836 texts altogether. When we present accuracy scores, they are based on a 10-fold cross-validation.

We use two different kinds of feature groups: data- and theory-driven. We have made this distinction because of the different approaches inherent in each. Namely, our theory-driven features are hand-engineered; we are checking for specific linguistic units or ratios we have theorized to predict proficiency. As for data-driven features, we are looking at the data as a whole, analyzing what we find by a given feature extraction method, without any concrete hypotheses. Our data-driven features are n-grams, parse rules, and grammatical tags, while our theory-driven features can be categorized into traditional, lexical, frequency, morphological, syntactic, and error measure sets. While this distinction is our own contribution, the specific features in the groups are mostly re-implementations of the features from Hancke’s thesis (2013). The following is a general description of the sets, pointing out some important differences from her work.

3.2 Theory-driven Feature Sets

- **Traditional Features** predate advanced machine learning techniques. They are based on surface-level features, such as the average number of characters per word. While Hancke (2013) only works with text length, and the average number of words and syllables, we experiment with a wide range of traditional formulae, such as the Flesch reading-ease score or the SMOG score.
- **Lexical Features** measure the range and variety of vocabulary used in a text by a writer. A traditional measure is the type-token ratio (TTR). As the TTR is sensitive to text length, various mathematical corrections of the original formula have been proposed, such as the root TTR, corrected TTR, log TTR, Uber Index and Yules K. More recent attempts to account for this problem include for instance the hypergeometric distribution diversity (HD-D) (McCarthy and Jarvis, 2007) and the measure of textual lexical diversity (MTLD) (McCarthy and Jarvis, 2010). It is an interesting counterpoint to mention the work of Treffers-Daller et al. (2018), who claim that

basic measures explain more variance in the CEFR levels of language learners' texts than the HD-D and MTLD, provided text length is kept constant across texts. Other lexical features are lexical density, measuring the ratio of lexical words to all words and lexical variation with respect to specific syntactic categories.

- **Frequency features** are based on the general idea that words that are more common in one language are acquired more easily and earlier by language learners. However, research also shows that especially L2 language learners often start with infrequent, more specific words (Crossley et al., 2011). We used a list with the number of occurrences of words in movie subtitles compiled by Brysbaert et al.(2011) to calculate the mean $\log(frequency)$ and the standard deviation of the $\log(frequency)$. The list was selected as subtitles are a common and easily available means of portraying everyday language. Hancke (2013) does not explain the choice of her binning method, so we chose equal width binning with 14 bins to determine whether words in certain frequency bands are characteristic of a specific level of text.
- **Morphological Features** are often realized through use of several linguistic features such as gender, case markers, verb tense markers, prefixes and suffixes. German is considered to be a morphologically rich language due to its three genders (masculine, feminine, neuter), four cases (nominative, genitive, dative, accusative), verb prefixes (both separable, such as *auf-*, and inseparable, such as *ver-*), and word compounding. In order to extract morphological features from the data set, we used the RFTagger (Schmid and Laws, 2008). For compound word detection, the CharSplit module for German was implemented (Tuggener, 2016).
- **Syntactic Features** measure the complexity of the dependency and parse tree structure of the text, based on Hancke (2013). She adapts these from various sources and fields and also adds some German-specific concepts to the feature set, such as the number of infinitival phrases with *zu*, or the ratio of passive constructions. For parse tree complexity,

examples include the length of production units measured by average length of sentences and clauses, the number of clauses per sentence, or ratios of dependent clauses, coordinating conjunctions, and complex nominals per clause and sentence. In addition, we have also included the ratio of separated verb prefixes. As for dependency features, a representative feature is for instance the maximum and average number of words between a head and a dependent in a text.

- **Error Measures** As we are dealing with data written by learners of the language, we have implemented a spell check that also counts the number of misspelled and corrected words. We use this number to calculate the ratio of misspelled words and total number of words. Implicitly the error measures are part of some of our other feature groups as well, for instance the RFTagger uses the tag *FM* ('foreign word') for words it does not recognize as German, and in our application, many of those would actually be misspelled words.

3.3 Data-driven Feature Sets

- **Parse Rule Features** Following Hancke (2013), Briscoe et al. (2010), and Yannakoudakis et al. (2011), we build a feature vector out of Parse Rule frequencies for each text. An example would be 'NP ART NN' standing for a Noun Phrase consisting only of an article and a noun.
- **N-grams** are a theoretically simple yet powerful set of features to extract from unstructured data, which are used in a wide variety of NLP tasks, as the words used in a text intuitively convey a lot of information about its makeup. In the field of automatic test scoring, Yannakoudakis et al. (2011) and Briscoe et al. (2010) have worked with them. While unigrams are powerful, they are not capable of handling phrases, but it is easy to improve them by adding bi- and trigrams to the feature sphere. In this project, we implemented word, lemma, character and POS n-grams.
- **Grammatical tags** are extracted with the RFTagger. Some examples are the type of particles (answer, degree, negation,

zu, separated verb particle) or the type of conjunctions (comparative, coordinating, subordinating with finite clause, subordinating with infinitive). Note that Hancke (2013) does not work with n-grams or grammatical tags.

4 Dataset

In order to overcome the limited availability of appropriate data, we use a combined data set built from five different sources: MERLIN (Wisniewski et al., 2011), Falko (Reznicek et al., 2012), KanDeL (Vyatkina, 2016), CLEG13 (Maden-Weinberger, 2013) and data from online sources for learners of German.¹ While all data sets contain different annotations, for the purposes of this project, only the CEFR level of the learner and the raw text were considered. As for the reading materials, we have decided to include them, as according to Pilán et al. (2016), textbook data can be beneficial for proficiency assessment in the event of a lack of data from the same domain.

The MERLIN corpus consists of texts written in an exam setting, which are assigned levels A1-C2 of the CEFR by trained human examiners. KanDeL is a collection of texts written by students from the US, who are enrolled in a basic German language program. The Falko corpus consists of text summaries written by C1-C2 learners of German and essays written by upper intermediate and advanced learners in various international institutions. Learners who scored more than 80 points on the C-test were hand-selected to form part of our C-level instances. The CLEG13 texts are essays, summaries and critical commentaries, and were written by students from the UK and labelled according to the year group of students into three groups. In the first two, students are assumed to be at levels B1 and B2, while the third consists of C1 learners.

See a summary of the combined data set in Table 1.

We are aware that the labels A, B and C do not necessarily signify the exact same level within the subcorpora. We have studied the levels' official description (Council of Europe, 2001) and the human-graded essays, and noticed that there can be significant differences inside one level. Thus, we believe that the categories are wide enough to

¹german.net/reading/, lingua.com/german/reading/, www.cornelsen.de/shop/capiadapter/download/get/

allow for the potential differences caused by the non-uniform labeling methods.

5 Results and Discussion²

5.1 Theory-driven features

When calculating all our theory-based features, we arrive at a total of 129. See Fig. 1 for a PCA (Principal Component Analysis) graph of the data set. PCA is a method for dimensionality reduction of data by retaining as much variance (information) as possible. It is easy to note that while A and C are neatly separated, level B is more interspersed throughout the graph. This result is intuitively plausible: While it is easier to give a casual definition for a beginner or an advanced speaker, the intermediate level, by its very definition, is a less well-defined category in between the two.

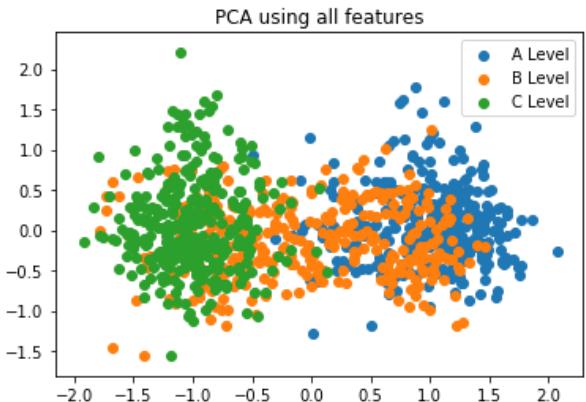


Figure 1: PCA with our theory-driven features

Our results are not directly comparable to any of the literature we have encountered, mostly due to the different class labels used. Hancke (2013), our main background literature, classifies according to A1-C1 on the MERLIN data set. We have re-implemented a large part of her best performing features, which she calls Best34 (Hancke, 2013, p. 56). She achieves 72.5% accuracy and we achieve 69% in the same setting. Our assumption is that the difference is due to the different spell checker we use, as the MERLIN data is very noisy compared to our other data sources. Hancke (2013) uses Google Spell Check³, which was not publicly

²One can argue that there is a conceptual overlap between the theory- and data-driven feature sets. However, the two feature sets are kept distinct throughout the experiments so it should not affect the results.

³<https://code.google.com/p/google-api-spelling-java/>

	Level A	Level B	Level C
CLEG13		416 (172,876)	315(146,545)
FalkoSummaries			107 (40,787)
FalkoEssay			159 (84,519)
FalkoWHIG			92 (56,513)
KanDeL	185(29,635)		
MERLIN	363(22,576)	624 (103,986)	
Reading	64 (10,390)	41 (8,642)	
Total	612(62,601)	1,081(285,504)	673(328,364)

Table 1: Number of texts (and tokens) in the subcorpora

Feature set	Data Set	Classes	Acc.
Best34	MERLIN	A1-C1	72.5
Best34 by us	MERLIN	A1-C1	69
Our best	MERLIN	A1-C1	70
Our best	Combined	A-C	82

Table 2: Results

available at the time of our project. Using our best features on the same data in the same setting, we achieve an accuracy of 70%. For a comparison, see a list of our best features in Table 3 and Best34 in Hancke (2013, p. 56). Testing on our newly created dataset, the model achieves a 82% accuracy, and most importantly, it very rarely misclassifies A for C, or vice versa. The verification of this statement is shown in the confusion matrix in Figure 2. The figure also shows that the model makes the highest number of incorrect predictions when classifying level B. See a more detailed error analysis in section 5.3, and a summary of the results in Table 2.

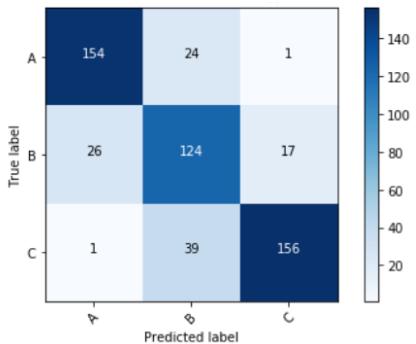


Figure 2: Confusion matrix presenting the classification results on a test set.

As for the analysis of our feature groups, when tested alone, traditional features, lexical features, morphological features and syntactic features

achieve an accuracy above 70%, with morphology being the best group. The reason for that might be that German is a morphologically complex language, and also that some of our morphology features capture the complexity of other fields as well. For instance the different verb forms, while morphologically different, also represent syntactic complexity. Even the spelling error features alone achieve an accuracy that is significantly better than the random baseline of 0.33.

When performing an iterative feature elimination, we arrive at around 40 features that perform approximately equally as well as our whole feature set. Additional features do not increase the classification accuracy significantly. The 40 best consist of 5 traditional, 8 lexical, 2 frequency, 5 spelling error, 14 morphological and 6 syntactic features.

These features are similar to Hancke's (2013) best-performing group, with the main difference being that she does not work with the traditional readability features, and we are not using language model features. See the features that performed best in this project in Table 3.

It is interesting to note that all of our spelling error features are in the best group, signaling that analysis of the errors the writer makes is a good direction for future additions to the feature set. While working with the data, we have noticed that the frequency of spelling errors noticeably changes across corpora. Ideally, the setting in which the text was written should also be taken into account, as the MERLIN texts which contain the most spelling errors were written in an exam setting, while other corpora also include homework assignments.

Inside the syntax group, we can see that the general complexity features perform well, e.g., average number of words between the head and a dependent, average clause length, number of

Group	Features
Traditional	SMOG, average number of characters per word, number of polysyllables, FOG
Lexical	Lexical Diversity Yule's K, Uber Index, HDD, MTLD
	Lexical Density and Variation adverb variation, modifier variation, verb variation, corrected verb variation
Frequency	bin 0, bin 6, mean frequency
Error	spelling errors, spelling errors with correction, capitalization errors, umlaut spelling errors, real spelling errors
Morphological	number of articles, ratio of compound nouns to noun, number of 1st person tags, number of past tense tags, ratio of nominative to nouns, number of nominatives, ratio of <i>-keit</i> suffix to nouns, number of past participle verbs, ratio of participles to verbs, number of singular tags, ratio of dative nouns to nouns, number of second person tags, ratio of verbs per sentence, ratio of 1st person to finite verbs
Syntactic	Dependency average number of words between head and dependent, average number of dependents per noun excluding modifiers
	Parse Tree Complexity average clause length, average number of dependent clauses per clause, average number of non-terminals per sentence, average number of interrogative clauses per sentence

Table 3: Best-performing features

non-terminals, or the ratio of dependent clauses per clause. From the more specific features we can see that NP complexity is relevant.

As for morphology, we can see that both compounds and derivational features (nouns ending with the suffix *-keit*), as well as inflections appear in the best-performing features. Certain verb forms, like past tense, participles, 1st and 2nd person have a correlation with proficiency. The inflection of nouns is also relevant, and the number of datives and nominatives divides the data best.

One can see that a lot of the features are dependent on sentence length, e.g. the traditional readability measures SMOG and FOG, number of polysyllables or most of the lexical features. The degree to which sentence length influences the classification level is up for debate. While it is true that it is theoretically possible to produce a short but complex or a long but simple text, in real life scenarios text length and complexity or proficiency very often go hand in hand. Exploring this correlation further is a direction for further research.

5.2 Data-driven features

We have experimented with word, lemma, character and POS n-grams. For words and

lemmas, instead of a simple count, a tf-idf (term frequency-inverse document frequency) weighting method is used, with the goal of scaling down the effect of features that occur very frequently and are thus not informative, for instance *the*. We set the length of n-grams to 3 to avoid the problem of data sparsity. We have iteratively experimented with the number of features that gives the best accuracy.

According to, for instance, the findings of Zesch et al. (2015), when analyzing the task-dependency of features, n-grams are highly task-dependent. Thus, as expected, our n-gram model performs rather poorly when trained on one subdataset and tested on another. See Fig. 3 to observe how closely n-grams are related to the data set and specific tasks that the learners were writing about: Words such as *Kansas* show up since one corpus is written by students from Kansas, as well as *Feminismus* ('feminism'), as that was one of the essay topics. Some features are more generalizable, for instance *ich* ('I') is an important feature for A level text, which relates to the communicative skills needed for beginner levels, i.e., they are expected to be able to talk about their immediate surroundings. In fact, all of the words support this observation. Inside the negative coefficient group, we can see the complementiser *dass* ('that'), showing a syntactic

feature; low-level learners are likely to not use dependent clauses.

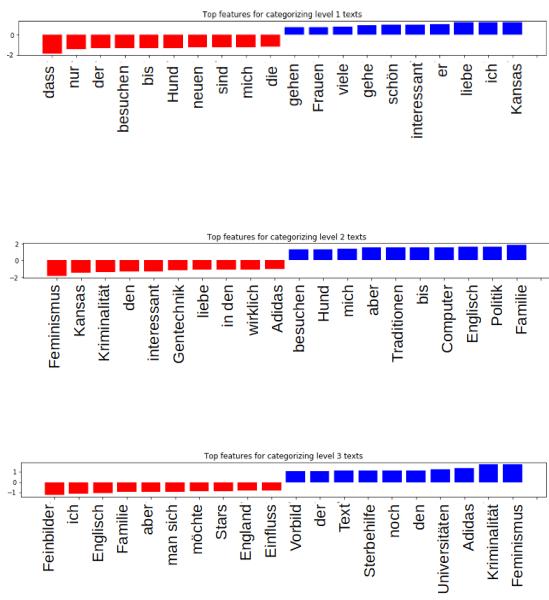


Figure 3: Important word n-grams

POS-unigrams, while producing a low, 59% accuracy, do present some interesting observations. For level A, some of the most predictive features are the presence of FM (foreign material), probably due to misspellings, VFIN (finite verbs), or INT (interjections). The VINF (infinitival verb) shows up in the features for levels B and C, as they are part of more complex verb structures, and the use of a PROADV (pronominal adverb) or VPP (participle verb) signals a level C.

See Table 4 for the accuracies of our different n-gram features. Trained on a specific task, they could achieve really high accuracy and we notice interesting observations looking at their results, however, they generalize poorly. Note that the cross-dataset accuracy is binary.

N-gram	#Feat.	Accuracy	Cross-data
Word	10,000	0.756	0.66
Lemma	5,000	0.748	0.63
Char.	20,000	0.881	0.55
POS	5,000	0.835	0.65

Table 4: Accuracy of n-gram features

The parse rules (PR) are extracted by the Stanford Parser. In order to reduce computational complexity and increase relevance, we have

excluded rules that appear fewer than 10 times in the data set. With this we arrived at 1222 parse rules. The length of the parse rules can be anything greater than or equal to two. We have achieved the best accuracy with 200 PR features, which was 0.766 ± 0.056 . See Table 5 for some of the best-performing PR features and possible interpretations. The best accuracy we have reached is 77% with 500 PR features. We notice that the data-driven features support and validate our theoretical feature engineering. NP and PP complexity seems to be important in classification, as is the use of conjunctions and *zu*-infinitives. The table is intended as an illustration of possible interpretation of some rules with high importance. For an exact understanding of the PR features, the TIGER Treebank (Smith, 2003) can be consulted.

Interpretation	Parse rules
NP complexity	NP PIAT NN NP ADJA NN NP ART NN NP ART NN
PP complexity	PP APPR ART ADJA NN PP APPR NN
Conjunctions	CNP NN KON NN CAP ADJA KON ADJ
Adj. and adv.	AVP ADV ADV AP ADV ADJD
Zu-Infinitive	VZ PTKZU VVINF

Table 5: Important Parse Rules and their interpretation

Prediction using the 85 grammatical tags of the RFTagger gave an accuracy of 79 (+/-4) %. The tags name, masculine, full verb, noun and coordinating conjunction are the best predictors for level A; attributive adjectives, personal pronouns, prepositions, and degree particles signal B level texts the strongest, while C level texts are best recognizable by colons, interrogatives, adverbs, negations and definite articles, according to the model. The presence of the word *zu* (in English corresponding to 'for', 'to' or the intensifier 'too') is the clearest sign of a text not being A level. We can conclude that in the case of this data-driven feature set as well, many of the features the system found to be important are the same as those we manually created. Some additional features, such as different kinds of articles, pronouns, or particles can be added to the model for future experimentation.

5.3 Error analysis

In order to think about directions for improving our classifier in the future, we have performed an error analysis on incorrectly classified sentences.

As we are dealing with a 3-way classification, the biggest error the classifier can make is a miss of two levels, e.g. classifying A instead of C. Running a classification 10 times and observing and analyzing these errors, we can state that the number is very low, ranging between 0 and 3 at each trial. An example sentence from one of these texts is *Dieser Film handelt die amerikanische revolutionäre Zeiten während der frühen Monate Jahres 1776, die auch ihm seine Name gibt.* (This film is about the American Revolutionary Times during the early months of 1776, which also gives it its name.) The text is part of the KanDeL data set, which includes both in-class assignments and homework. Our assumption is that the text was written as a homework assignment, so the writer could put time and effort into performing beyond their expected proficiency level. When observing the feature values for the text compared to the mean values for A texts, not only does it surpass them in the surface-level categories, but also in categories such as average number of words between head and dependent. Another example of misclassification is when there is not enough information in the text, for instance the A level text *LIEBER JENS, GLÜCKWUNSCH* ('DEAR JENS, CONGRATULATIONS') gets classified as C by our classifier. Due to its length, the relatively long term *Glückwunsch* or the high TTR are possibly given too much weight as many of the other features would be zero. Another text that showed up multiple times in this non-exhaustive experiment was a C-level text from the CLEG data that was classified as A. On closer inspection, comparing the feature values to the mean, the problem is the surface-level features, like number of syllables or SMOG. The implications of these findings is that the traditional readability formulas are not without their problems, which we are aware of. However, excluding them completely from the calculation is not the best option, as they show up in the most distinctive features.

The question of errors by one level is more complicated. We can conclude that texts from the CLEG data set are mostly classified higher than their label. This might be unintuitive as in the case of CLEG, the labels are actually the level the

students are supposed to be at a certain academic year, and not dependent on any test score or essay grade. Working with the data, we have already noticed that CLEG level B is closer to the level C from other sources. Texts from the MERLIN data are the most commonly misclassified in both directions. We assume that level A texts often get misclassified because of the lack of information they contain. We also have to note that our classifier at the moment does not take the flow or cohesion of texts, or correct word choice, into account, which would probably be vital when dealing with such a short text; looking at the feature vector, we see that such a text tricks our classifier in terms of surface-level and lexical features which are highly correlated with text length.

6 Conclusion

With the help of our classifier and the data set, the writing level of language learners can be found with a reasonably high accuracy. We found that linguistic features correlate with CEFR proficiency levels and can perform reasonably well in a classification scenario. Moreover, with our detailed description of the performance of different features, we hope to have come closer to a tool that helps educators obtain a more practical list of what is expected from learners at certain levels. In the case of German, our target language, morphological features appear to be especially important. Some syntactic and lexical features are also given a high weight by the machine learning algorithm.

By constructing a larger and more balanced data set, we report 82% accuracy, a significant improvement over our models performance on just the MERLIN texts, which reached 70%. For further investigation, the most important factor is the data set itself. With enough data, we can also try running the model on the full CEFR scale from A1-C2, instead of the three-level classification currently being performed. Additional improvements to the data preprocessing can also be incorporated into our current pipeline, such as experimenting with different German spell checkers and sentence boundary detection methods. As for the feature groups, data from other fields, such as semantic or pragmatic information, are not included in the scope of this project; this additional information would also be worth testing. In other cases, a more fine-grained feature division could

be helpful, for instance, analyzing different error categories.

It is also important to note that the feature sets, while tailored for German, are not language-specific per se. The data-driven features are all language-neutral and as for the theory-driven ones, traditional, lexical, frequency, and error measures are also not tied to the language of the text. As German is a morphologically rich language, it is unsurprising that morphological features perform well for classification, which may not be the case for other, morphologically less rich languages. As for syntactic features, most features are related to the dependency or parse tree structure and thus, also language neutral. However, a few features, such as the number of passive constructions, or specific infinitival phases we would not expect to contribute to the results greatly in other languages. Testing cross-language performance is a promising direction for future research, as well.

References

- T. Briscoe, B. Medlock, and O. Andersen. 2010. Automated assessment of ESOL free text examinations. Technical report, University of Cambridge Computer Laboratory.
- M. Brysbaert, M. Buchmeier, M. Conrad, A. Jacobs, J. Bölte, and A. Böhl. 2011. The word frequency effect: A review of recent developments and implications for the choice of frequency estimates in german. *Experimental Psychology*, 58:412–424.
- X. Chen and D. Meurers. 2016. CTAP: A web-based tool supporting automatic complexity analysis. in proc. of the workshop on computational linguistics for linguistic complexity.
- Council of Europe. 2001. *Common European Framework of Reference for Languages: Learning, Teaching, Assessment*. Press Syndicate of the University of Cambridge.
- S. A. Crossley, T. Salsbury, D. S. McNamara, and S. Jarvis. 2011. Predicting lexical proficiency in language learner texts using computational indices. *Language Testing*, 28(4):561–580.
- J. Hancke, S. Vajjala, and D. Meurers. 2012. Readability classification for german using lexical, syntactic, and morphological features. In *Proc. of COLING 2012: Technical Papers*, pages 1063–1080.
- J. Hancke. 2013. Automatic prediction of CEFR proficiency levels based on linguistic features of learner language. Master’s thesis, University of Tübingen.
- R. Lavalle and K. Berkling. 2014. Data exploration of sentence structures and embellishments in german texts: Comparing childrens writing vs literature. In *Proc. of the 12th edition of the KONVENTS conference, Hildesheim*, volume 1.
- U. Maden-Weinberger. 2013. CLEG13 corpus of learner german. documentation.
- P. McCarthy and S. Jarvis. 2007. vocd: A theoretical and empirical evaluation. *Language Testing*, 24:459–488, October.
- P. McCarthy and S. Jarvis. 2010. Mtld, vocd-d, and hd-d: A validation study of sophisticated approaches to lexical diversity assessment. *Behavior Research Methods*, 42:381–392.
- A. Nietzio, B. Scheer, and C. Bühler. 2012. How long is a short sentence? - a linguistic approach to validation and definition of rules for easy-to-read material. In K. Miesenberger, A. Karshmer, J. Klaus, and W. Zagler, editors, *Proc. of Computers Helping People with Special Needs, 13th International Conference, ICCHP 2012*, pages 369–376.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- I. Pilán, E. Volodina, and T. Zesch. 2016. Predicting proficiency levels in learner writings by transferring a linguistic complexity model from expert-written coursebooks. In *COLING*.
- M. Reznicek, A. Lüdeling, C. Krummes, F. Schwantuschke, M. Walter, K. Schmidt, H. Hirschmann, and T. Andreas, 2012. *DasFalko-Handbuch. Korpusaufbau und Annotationen. Version 2.01*.
- H. Schmid and F. Laws. 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. *COLING 2008, Manchester, Great Britain*.
- G. D. Smith. 2003. A brief introduction to the tiger treebank, version 1.
- J. Treffers-Daller, P. Parslow, and S. Williams. 2018. Back to basics: How measures of lexical diversity can help discriminate between cefr levels. *Applied Linguistics*, 39(3):302–327, 04.
- D. Tuggener. 2016. *Incremental Coreference Resolution for German*. Ph.D. thesis, University of Zurich, Faculty of Arts.
- S. Vajjala. 2013. *Analyzing text complexity and text simplification: connecting linguistics, processing and educational applications*. Ph.D. thesis, University of Tübingen.

- T. vor der Brück and S. Hartrumpf. 2007. A semantically oriented readability checker for german. In *Proc. of the 3rd Language & Technology Conference, Poznan, Poland*, page 270274, October.
- N. Vyatkina. 2016. KANDEL: A developmental corpus of learner german. *International Journal of Learner Corpus Research*, 2(1):102–120.
- Z. Weiss and D. Meurers. 2018. Modeling the readability of german targeting adults and children: An empirically broad analysis and its cross-corpus validation. In *Proc. of the 27th International Conference on Computational Linguistics (Coling 2018)*.
- K. Wisniewski, A. Abel, and D. Meurers. 2011. Merlin. multilingual platform for the European reference levels: Interlanguage exploration in context.
- H. Yannakoudakis, T. Briscoe, and B. Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1 of *HLT '11*, pages 180–189, Stroudsburg, PA, USA. Association for Computational Linguistics.
- T. Zesch, M. Wojatzki, and D. Scholten-Akoun. 2015. Task-independent features for automated essay grading. In *Proc. of the Building Educational Applications Workshop at NAACL*.

A Probabilistic Morphology Model for German Lemmatization

Christian Wartena

Hochschule Hannover

Expo Plaza 12, 30539 Hannover, Germany

christian.wartena@hs-hannover.de

Abstract

Lemmatization is a central task in many NLP applications. Despite this importance, the number of (freely) available and easy to use tools for German is very limited. To fill this gap, we developed a simple lemmatizer that can be trained on any lemmatized corpus. For a full form word the tagger tries to find the sequence of morphemes that is most likely to generate that word. From this sequence of tags we can easily derive the stem, the lemma and the part of speech (PoS) of the word. We show (i) that the quality of this approach is comparable to state of the art methods and (ii) that we can improve the results of Part-of-Speech (PoS) tagging when we include the morphological analysis of each word.

1 Motivation

In the following we present a simple approach to lemmatization of German texts and compare the results with a number of other easily available tools.

The motivation was twofold: while lemmatization seems to be a core task in analyzing text, in most standard Python packages like Stanford's Natural Language Toolkit (NLTK), no lemmatization for German is available. Mainly for teaching undergraduate students we wanted to have a simple tool, that gives linguistically correct lemmata for all words and is also easy to use and install on a Python notebook server. In the second place, we wanted to investigate, whether a careful splitting of a word into a stem and suffix can improve the treatment of unknown words in a standard trigram PoS tagger, in which the PoS otherwise is guessed on the base of the final letters of a word.

Lemmatization is a core task in analyzing text. Nevertheless it did not receive as much attention as e.g. PoS tagging. Rule based systems can reach

a very high accuracy for morphological analysis. However, existing systems are not always available and the construction of a lexicon and morphological rules is a very tedious task. In practice therefore often simple heuristic rules or a dictionary lookup are used, even if the quality of the tools is not even known. Especially for information retrieval the quality of lemmatization seems not to be very important and some studies suggest that any form of heuristic stemming, mixing up inflectional and derivational morphology can be used (Kettunen et al., 2005; Moral et al., 2014) though some other studies contradict these findings (Braschler and Ripplinger, 2004).

In the following we present an approach to morphological analysis based on computing the most likely sequence of morphemes for a given word. In section 2 we present the details of this method. In subsection 2.4 we show how we can use the results in a PoS tagger. In section 3 we discuss related work and alternative approaches and finally, section 4 compares the results on lemmatization and PoS tagging.

2 Method

Given a word $w = a_1 \dots a_n$ we try to find the most likely sequence of morpheme tags $s = t_1 \dots t_k$ that generates w . We cannot use a standard Hidden Markov Model and the Viterbi algorithm to find the most likely sequence s since we do not have a segmented list of output observations. However, the solution presented here is very similar to a Hidden Markov Model and the computation of the most likely tag sequence is almost identical to the Viterbi Algorithm. We define the most likely tag sequence s that generates w as

$$\max_{k,s \in T^k} \prod_{i=3}^k p(t_i | t_{i-2}t_{i-1}) \cdot p(a_{l_i m_i} | t_i) \quad (1)$$

where T is the set of all tags, $0 \leq l_i \leq m_i \leq n$ for each i and $w = a_{l_3 m_3} \dots a_{l_k m_k}$.

Since here every state is dependent on two previous states we call the model second order model and we have to add two start states to every sequence. We also add a final state that generates the empty string to each tag sequence. We add one final state for each PoS tag. This will allow us, to compute the most likely tag sequence for each PoS.

Using dynamic programming we can find the optimal tag sequence efficiently. Using a first order model, for a string $w = a_1 \dots a_n$ we define the probability that a prefix $a_1 \dots a_j$ of w is tagged with a tag sequence in which the last tag is t as

$$\vartheta(t, j) = \max_{r, s, i} (\vartheta(s, i) \cdot p(a_{ij} | t) \cdot p(t | s)) \quad (2)$$

for each $t \in T$ and $2 \leq j \leq n$ where $s \in T$, $2 \leq i \leq j$, a_{ij} denotes the substring $a_i \dots a_j$ from w and

$$\vartheta(t, 1) = p(t | \text{START}). \quad (3)$$

The equation can easily be extended for a second order model but the becomes slightly more complicated. When we compute $\vartheta(t, j)$ we get the well-known Trellis diagram. When we extend the algorithm with a backpointer, we can easily find the optimal tag sequence for a given word.

Consider e.g. the word *Sorgen*, that can either be the plural of the noun *Sorge*, the infinitive of the verb *sorgen*, or a finite form of the same verb. Now, for each of the corresponding final states we can compute the most likely tag sequence that generates the word *sorgen*. In our data we thus find the following tag sequences:

$$\begin{aligned} s_{\text{inf}} &= \text{None, START, VV, SUF_INF, END_VVINF} \\ s_{\text{fin}} &= \text{None, START, VV, SUF_FIN, END_VVFIN} \\ s_{\text{nn}} &= \text{None, START, NN, SUF_NN, END_NN} \end{aligned}$$

The probability that the sequence s_{nn} generates the word is computed as follows: $p(\text{Sorgen}, s_{\text{nn}}) = p(\text{NN} | \text{None, START}) \cdot p(\text{SUF_NN} | \text{START, NN}) \cdot p(\text{END_NN} | \text{NN, SUF_NN}) \cdot p(\text{'sorge'} | \text{NN}) \cdot p(\text{'n'} | \text{SUF_NN}) = e^{-1.60854} \cdot e^{-1.46985} \cdot e^{0.0} \cdot e^{-7.82129} \cdot e^{-1.43789} = e^{-12.33757}$. Similarly, we find $p(\text{sorgen}, s_{\text{fin}}) = e^{-10.77681}$ and $p(\text{sorgen}, s_{\text{inf}}) = e^{-10.63024}$.

Since there could be several sequences generating *sorgen* and ending in `END_NN`, the probability $p(\text{Sorgen}, s_{\text{nn}})$ is not the probability that the word is generated by a noun sequence. To compute that probability we would need an equivalent of the forward algorithm, that computes the sum of all probabilities leading to one state instead of the maximum.

However, in practice alternative paths turn out to be completely nonsense (since the model is highly over generating) or extremely unlikely (and thus do not change anything).

Given the large amount of training data for words, and the fact that for each substring we can assume that it is generated by one of the open class morphemes, we do not need any interpolation or smoothing and use the trigram probabilities directly.

Finally, we use also case information and multiply the found probability with the probability that a word of the found class is capitalized or not.

2.1 Unknown Words

For each string we can assume that it is an unseen instance of an open morpheme class. Currently we defined by hand, which classes are the open classes, but this also can quite easily be guessed from the number of hapax legomena in each class.

For a morpheme $m = a_1 \dots a_n$ we can estimate the probability that an unseen morpheme is generated by a given morpheme tag using the probability that a morpheme ending on a given suffix is generated by that class. We compute these suffixes probabilities on infrequent morphemes, assuming that morphemes not observed in the test data are more similar to infrequent than frequent to morphemes. If not enough observations are available for suffixes of length n we use the probabilities for suffixes of length $n - 1$. To compute the probabilities of the shorter suffixes we exclude all morphemes ending on one of the longer suffixes for which we had enough observations. E.g. if we use the probability $p(\text{noun} | \text{ung})$, for bigrams we use $p(\text{noun} | \text{ng and not ung})$ rather than $p(\text{noun} | \text{ng})$.

For longer unknown words we need to be sure, that an analysis using several known morphemes is preferred over the analysis as one unknown morpheme. Especially long nouns should be much more likely to be a noun compound, consisting of two or more known stems than being a completely unseen stem. Thus we also use the probability $p(n | t)$ that a morpheme of length n is generated by t . We compute this probability on infrequent morphemes again. Finally, we use the probability $p_{\text{hap}}(t)$ that the tag produces a hapax legomenon. Thus we approximate the probability of an unseen morpheme $m = a_1 \dots a_n$ given a tag t as

$$p(m | t) \approx p(a_{n-2}a_{n-1}a_n | t) \cdot p(n | t) \cdot p_{\text{hap}}(t). \quad (4)$$

2.2 Generating training data

The most critical part in the development of the analyzer is the generation of training data. We generate the training data from the Tiger Corpus (Brants et al., 2002). Here we find a lemma and a PoS for each word form. The basic idea now is to split the word form in the stem, that can easily be derived from the lemma, and prefixes and suffixes. E.g. the word *geplant* with lemma *planen* and stem *plan* can be split up in *ge*, *plan* and *t*. For the affixes we assign tags based on the given PoS. Thus, in the present example we generate

```
[('ge', 'PREF_PP'), ('plan', 'VV'), ('t', 'SUF_PP'), ('', 'END_VVPP')]
```

In many cases we end up with much more complicated sequences. We restrict the decomposition of words to inflectional morphology except for noun compounds, comparatives and superlatives of adjectives and adjectives derived from participles. Thus we have sequences like

```
[('auf', 'PTKVZ'), ('ge', 'PREF_PP'), ('schreck', 'VV'), ('t', 'SUF_PP'), ('', 'END_VVPP')]
[('amtier', 'VV'), ('end', 'PRESPART'), ('en', 'SUF_ADJ'), ('', 'END_ADJA')]
[('ordnung', 'NN'), ('s', 'FUGE'), ('kräft', 'NN_VAR'), ('en', 'SUF_NN'), ('', 'END_NN')]
```

We use several language dependent heuristic rules to split up each word. In German the stem often is not a part of the surface form. In most of these cases we can find a variant of the stem by searching a substring that starts and ends with the same consonants. E.g. for the word *jünger* (younger) with stem *jung* (young) we find:

```
[('jüng', 'ADJ_VAR'), ('er', 'ADJ_COMP'), ('en', 'SUF_ADJ'), ('', 'END_ADJA')]
```

In addition now the substitution *jüng/jung* will be stored for the adjective class. These substitutions will be used later to reconstruct the stem and lemma of an analyzed word.

In total we used 52 final tags (i.e. tags encoding the PoS of a word and not corresponding to any morpheme) and 75 real morpheme tags.

The morpheme classes obtained in this way are very rough and result in a massively overgenerating model. E.g. for some verbs the past participle is formed without the prefix *ge*. Thus the model allows the morpheme tag *SUF_PP* without having seen the morpheme *PREF_PP* before and independent of the verb, since no distinction is made between verb classes needing the prefix and those

that do not have this prefix in the past participle. Thus even a form like *lauft* could be analyzed as [('lauf', 'VV'), ('t', 'SUF_PP'), ('', 'END_VVPP')]. However, it turns out that for analyzing there is only a limited number of cases, where this causes problems.

2.3 Lemmatization

Once the most likely sequence of tags is found, a small set of rules is used to generate the correct lemma. These rules mainly deal with the generation of an infinitive from a stem and with the application of the stored substitutions for irregular stems and stems with Ablaut (vowel gradation).

2.4 Part of Speech Tagging

The usual way to analyze German or English is to start with part-of-speech tagging and then to analyze each word according to the found PoS. It is now tempting to investigate, whether the other way around works as well: instead of using observed probabilities we could use the probabilities as computed by the morphological analysis. To be precise: we computed $p(w, t)$ for a word and a tag before. In a standard trigram tagger (see e.g. Brants (2000)) we need the probability $p(w | t)$. This probability can be computed easily by using the fact that $p(w | t) = \frac{p(w, t)}{p(t)}$.

The approach has the advantage that we get much better statistics for inflectional variants of infrequent words. On the other hand we lose a lot of information on specific word forms. For some words only certain forms are frequently used, and others are infrequent or even not existent. E.g. the noun *Ärger* (trouble, annoyance) does not have a plural form. Consequently, the form *ärgere* only can be a verb form (from *ärgern*, to annoy). Using the morphological analysis described above, we will nevertheless find an analysis as noun as well.

In the following we will use two variants: in the first variant we use only the probabilities computed by the morphological analysis. In the second variant we will use the observed probabilities and use the morphological analysis for words that were not observed in the training data. Here we treat words seen once and twice as unseen words as well.

We base the transition probabilities on trigram statistics over tags. Here we use linear interpolation to avoid zero probabilities and set the smoothed probability $p^*(t_n | t_{n-2}t_{n-1}) = 0.95 \cdot p(t_n | t_{n-2}t_{n-1}) + 0.04 * p(t_n | t_{n-1}) + 0.01 \cdot p(t_n)$.

2.5 Implementation

We implemented all algorithms in pure Python. The script to generate training data from the Tiger corpus and the classes to train and apply the morphological analysis and the PoS tagging are available on Github¹ and PyPI².

In order to speed up the computation, we compute the analysis for 2000 frequent words immediately after training and store the results in the model file as well. Here we exclude all analyses resulting in a PoS tag that was never observed for that word. This also slightly improves the results.

In the following we will call this tagger *Hanover Tagger* or short, *HanTa*, and refer to the version using observed probabilities for all words that were seen at least three times in the training data as *HanTa (hybrid)*.

In the package available on GitHub there are functions to analyze a single word, to tag a sentence or to tag and lemmatize a sentence at once. The user can also choose to get only the PoS tag, the lemma, or a morphological analysis.

3 Related Work and Alternative Tools

At first glance lemmatization seems to be an easy task. Nevertheless, for most languages, at least for German, we need some morphological analysis to find correct lemmata. State of the art methods for morphological analysis are still rule based. In the first place here the work of Koskenniemi (1983) has to be mentioned. For German this approach was used in the SMOR tool (Schmid et al., 2004).

Besides the rule based approaches there are several attempts to derive a morphological model for a language in a complete unsupervised way. An example of this approach is Morfessor (Creutz and Lagus, 2007), that in fact uses an underlying model for morphology that is very similar to ours. For a recent overview of unsupervised learning of morphology we refer to (Goldsmith et al., 2017).

Only a few studies deal with the possibility to learn lemmatization or morphology in general from annotated data. Kanis and Müller (2005) and Jongejan and Dalianis (2009) learn rules from a lemmatized corpus to transform an inflected word form to a lemma. Gashkov and Eltsova (2018) obtain good results for German by a full-form dictionary and applying analogy for unknown words: basically, for an unknown word form the word with

¹<https://github.com/wartaal/HanTa>

²<https://pypi.org/project/HanTa/>

the longest common suffix is searched and then the transformation associated with that word is applied. Gesmundo and Samardžić (2012) propose to annotate words with the type of rule, needed to transform the full form to a lemma, thus reducing lemmatization to a tagging task. A similar idea is followed by Chrupala et al. (2008) who define classes that correspond to mappings from word forms to lemmata and train a classifier to classify words accordingly. This approach is extended by Müller et al. (2015) who use more features and conditional random fields for classifying morphemes. To some extend our model resembles this approach. The main differences are (i) that we learn on segmented data (and thus have to produce such data before learning) and (ii) that Müller et al. (2015) learn the transformation needed to produce a lemma as well, while we need a small language specific, rule based component that produces a lemma from the list of morphemes found.

Our goal is not to improve on state of the art morphological analysis but just to have an easy tool that gives results that can be used in further tasks and to provide an alternative for lemmatization tools that are easily available and therefore used frequently. In the following we thus compare the results of lemmatization to those obtained by the TreeTagger, Spacy and GermaLemma. For testing PoS tagging we use the same tools and in addition an own implementation of a standard second order Hidden Markov Model, using suffix statistics to guess the output probabilities for unseen words.

The TreeTagger (Schmid, 1999) is a PoS tagger based on a second order Hidden Markov Model (or trigram model) extended with decision trees to use more contextual information and dictionaries of prefixes and suffixes to improve the basic model. The standard model for German was trained on a manually tagged newspaper corpus.

Spacy (ExplosionAI GmbH, 2019) is a state of the art tool based on deep learning for tokenization, PoS tagging and named entity recognition. Spacy also provides lemmatization. While other modules are based on trained artificial neural networks, lemmatization is rule based. We used release 2.1.4.

GermaLemma (Konrad, 2017) is a tool that combines a full form lexicon, extracted from the Tiger Corpus, an algorithm for splitting compounds and morphological rules from the Pattern package (The CLIPS (Computational Linguistics & Psycholinguistics) research center, 2018). GermaLemma

requires that the lemmatization is preceded by PoS tagging.

4 Evaluation

Since our focus is on the development of a practical tool for lemmatization, that can be used as a component in a larger pipeline, we will use large corpora, in which many words occur many times, instead of word lists for evaluation.

4.1 Data

As mentioned before we use the Tiger Corpus for training. We used version 2.2 which consists of $50 \cdot 10^3$ sentences or $0.9 \cdot 10^6$ tokens. For cross validation we split the corpus into 10 contiguous parts, as was also done by Giesbrecht and Evert (2009) and is considered to be a slightly harder and more realistic setting than taking every tenth sentence, since every part now gets sentences from different texts. In addition we use a list of the most frequent verb forms extracted from the DeReKo Corpus (Stadler and Wegstein, 2016) to train the morphology model and to make sure, that at least the most frequent verb stems are seen in the training phase.

For evaluation, besides the Tiger Corpus, we use TüBA D/Z and the Hamburg Dependency Treebank. The Hamburg Dependency Treebank (HDT) (Foth et al., 2014) is very interesting for our purpose since it consists of texts from a different domain. Tiger and TüBA D/Z consist of daily newspaper texts, while HDT uses texts from `heise.de` with news and background articles on anything related to computer hard and software. Here we observe the use of a different vocabulary and sometimes deviations from standard German spelling, like writing compounds as words separated by blanks. We use part A of the corpus, which was manually annotated and checked for consistency. This part consists of 102 000 sentences or $1.87 \cdot 10^6$ tokens. We use HDT for evaluation of PoS Tagging. The lemmata provided cannot be used for evaluation, since for compounds only the head is given as a lemma.

TüBA D/Z (Telljohann et al., 2004) is a manually annotated newspaper corpus of a similar size (104.787 sentences or $1.96 \cdot 10^6$ tokens). TüBA D/Z uses a slightly different tagset than Tiger: TüBA D/Z has a different tag for pronominal adverbs (which we just can replace to compare results) and it distinguishes between two different forms of attributive indefinite pronouns (with and

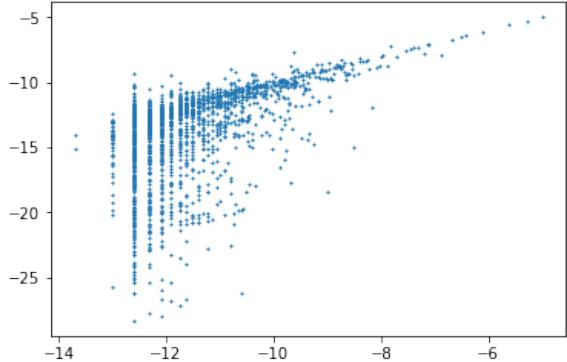


Figure 1: Observed (x-axis) vs. predicted probabilities (y-axis) (here displayed as the natural logarithm of the probabilities) for 2337 word-tag pairs.

without determiner) while Tiger just has one. Thus, we remove this distinction when evaluating results trained on the Tiger annotation scheme. With regard to the lemmata, TüBa D/Z uses a #-sign to mark the boundary of separable prefixes and sometimes adds disambiguating PoS information to the lemma. Both are removed. In some cases (especially for adjectival nouns) several possibilities for the lemma are listed and separated by a pipe symbol. Here we keep the whole string as it is.

4.2 Lemmatization

First, we compare the values of the predicted probabilities with the observed probabilities. For this purpose we take every 10th word of a list of all word forms occurring at least 3 times. This results in a list of 2337 words. For each of these words we compare the probability for the most probable observed tag with the probability estimated for that tag. The Pearson correlation between the two methods is 0.455 indicating a low correlation between the observed and predicted values. Especially for infrequent word forms the estimates are much too low (see Figure 1). This situation is not completely unwanted: we will predict non-zero probabilities for many word forms not present in the corpus. Consequently, some observed probabilities have to become smaller.

4.2.1 Quantitative Analysis

The morphological analysis gives a ranked list of possible PoS tags for each word. We use precision, recall and Mean Reciprocal Rank (MRR) to evaluate these rankings, computed for one fold from the 10-fold cross validation division of the tiger corpus. Here we do not take into account the words

Table 1: Mean Reciprokal Rank on the prediction of the PoS on 10% of the Tiger corpus, using 90% as training data. The prediction is only based on the morphological analysis, not taking into account information from surrounding words.

	all words	unknown words
HanTa	0.955	0.900
HanTa (hybrid)	0.962	0.900

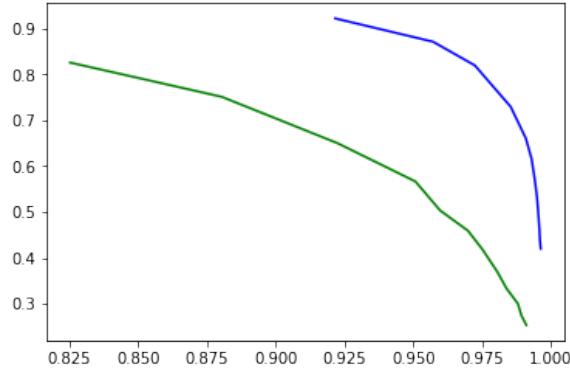


Figure 2: Precision and recall of the prediction of the PoS on 10% of the Tiger corpus. The green line gives the results for unknown words, while the blue (upper) line corresponds to all words.

around each word and for an ambiguous word we thus always predict the most likely tag.

Figure 2 shows the precision-recall curves for the tagger using only predicted probabilities. The MRR for known and unknown words for both variants of the tagger are given in Table 1.

Despite the low correlation of the observed and computed probabilities, the ranking of the results seems to be almost identical.

Next, we test the accuracy of lemmatization of HanTa, the TreeTagger, Spacy and GermaLemma on the Tiger and the TüBa Corpora. We use GermaLemma here in combination with our own Trigram Tagger implementation (GerTriTa). For GerTriTa and HanTa we use 10-fold cross validation on the Tiger Corpus. However, GermaLemma is trained on the Tiger Corpus. Thus, here we cannot really use the results. The same holds for Spacy that is trained on Tiger as well.

Since the correct lemmatization for many closed class elements is unclear and arbitrary (e.g. in Tiger the lemma of the determiner *das* is *der* while the TreeTagger generates the lemma *das*, which we do not want to consider as incorrect) we evaluate on

Table 2: Accuracy of lemmatization on the Tiger corpus. Values in brackets are obtained by evaluating on the training data

	all	unknown
HanTa	96.98 ±0.24	86.00 ±0.68
HanTa (hybrid)	97.12 ±0.25	86.00 ±0.68
TreeTagger	96.12	
Spacy	(87.46)	
GermaLemma	(97.79 ±0.20)	(96.38 ±0.37)

Table 3: Accuracy of lemmatization on the TüBa D/Z corpus.

HanTa	92.98
HanTa (hybrid)	93.06
TreeTagger	93.59
Spacy	86.60
GermaLemma	92.23

the open class words only. Results are given in Table 2.

We also evaluated the lemmatization on the TüBa D/Z Corpus. This corpus was not used in development or training of any of the compared tools (as far as we know) and therefore is much better suited for evaluation. The results are given in Table 3.

4.2.2 Error Analysis

For the HanTa lemmatizer we clearly see two main sources of errors. In the first place many plural forms of long (unknown) nouns are not correctly analyzed as a stem and a plural suffix. E.g. the word *Plattenläden* (Record shops), occurring in TüBa D/Z but not in Tiger gets the lemma *Plattenläden*, since the analysis as one long unknown word is slightly more probable than the analysis as a compound, which would have enabled HanTa to correctly lemmatize the word as *Plattenladen*. Also for a simple word like *Volkslieder* (Folk songs) HanTa preferred the analysis as one large unknown noun over the analysis *Volk+s+lied+er*. This is partly also caused by the quite low probability of the suffix *er*. Here it could help to have more fine grained classes that would give a higher probability for the suffix *er* after certain nouns.

The second source of errors is formed by adjectival nouns and especially present participles that are used as nouns, like *Lehrende* (*teaching person*). We did not code this type of nouns in any special

Table 4: Accuracy of PoS tagging on the Tiger corpus.

	all	unknown
HanTa	96.52 ± 0.33	88.98 ± 1.04
HanTa (hybrid)	96.96 ± 0.34	88.98 ± 1.04
GerTriTa	96.94 ± 0.31	88.04 ± 0.72
TreeTagger	95.08	

way in the training data, but just coded them as one nominal morpheme. In the Tiger corpus lemmata are not assigned uniformly to these type of nouns. E.g. the word *Andersdenkende* (dissenting person) is lemmatized as *andersdenkend*, the word *Asylsuchenden* (asylum seeking person) is lemmatized as *Asylsuchender* (with strong masculine flexion) and *Wohlhabenden* (wealthy person) as *wohlhabende* (with weak flexion). In the TüBa D/Z corpus these type of nouns have three lemmata (one for each gender), separated by a 'l'-sign in case the gender is underspecified and the lemma with the corresponding gender marking, if the gender is clear. Thus *Süchtigem* (addicted person, dative masculine singular) gets the lemma *Süchtiger*.

Most other lemmatizing errors are caused by ambiguity and the assignment of the wrong PoS. E.g. the word *überzeugt* (convinced) has to be lemmatized as *überzeugen* if it is a past participle, but it has to be lemmatized as *überzeugt* if it is a past participle used in an adjectival way (at least according to the annotation principles of Tiger and TüBa D/Z; see e.g. (Lenz, 1993) and (Eisenberg, 1994, p. 71) for a discussion on the status of German participles). Finally, the frequent words *möchte* and *möchten* (would like) are lemmatized incorrectly in Tiger as *möchten*, and thus learned incorrectly by HanTa, while they are correctly lemmatized as *mögen* in TüBa D/Z.

4.3 Part of Speech Tagging

For the evaluation of the PoS tagging based on the tag probabilities found by the lemmatizer we use two corpora: the TüBa D/Z treebank and the manually corrected part (part A) of the Hamburg Dependency Treebank. Especially, the latter one is interesting since its text are not from daily newspapers like the data from Tiger and TüBa D/Z.

The results for evaluating PoS tagging with 10-fold cross validation on the Tiger Corpus are given in Table 4. The results on TüBa D/Z and HTB are given in Table 5.

Table 5: Accuracy of PoS tagging on the TüBa D/Z and HDT Corpora.

	Tüba	HTB
HanTa	95.07	93.80
HanTa (hybrid)	95.54	94.29
GerTriTa	93.19	92.97
TreeTagger	94.81	92.87
Spacy	93.38	92.75

Table 6: Top 10 most frequent errors. The first column gives the correct PoS tag, the second column the predicted PoS and the last column the proportion this error has to the total number of errors.

PoS	Predicted PoS	Perc.
NN	NE	10.20 %
NE	NN	5.36 %
KOKOM	APPR	5.11 %
VVFIN	VVINF	4.23 %
NE	FM	3.09 %
ADV	ADJD	2.76 %
NN	ADJA	2.47 %
FM	NE	1.78 %
VVFIN	VVPP	1.69 %
KOUS	PWAV	2.63 %

4.3.1 Error Analysis

Finally, we have a more detailed look of the errors that HanTa makes on the TüBa-D/Z corpus. Table 6 shows the 10 most frequent errors. We see that there are some frequent ambiguous words, like *als* (as, than) and *wie* (as) that are hard to classify and already were reported by Giesbrecht and Evert (2009) to be a main source of errors. For most verbs the infinitive and first and third person present tense plural are identical and in many cases the correct class cannot be determined without syntactic analysis. Furthermore, there are many problems with proper nouns (NE). Here HanTa has difficulties to decide whether an unknown word is a proper noun (NE), a foreign word (FM) or a common noun (NN). In addition, Tiger and TüBa-D/Z also differ in the distinction between common nouns and proper nouns. E.g. the words *Osteuropa* (eastern Europe), *Bundesnachrichtendienst* (Federal intelligence office) and *EU-Kommission* (EU commission) are classified as proper names in Tiger but as common names in TüBa-D/Z.

Table 7: Average runtime of analyzing the first 1000 sentences from TüBa D/Z. All results are averages from 7 runs.

Tagger	Time
TreeTagger	2.85 s \pm 0.501 s
Spacy	14.5 s \pm 1.61 s
HanTa	10.7 s \pm 0.102 s
HanTa (hybrid)	6.94 s \pm 0.162 s
HanTa incl. lemm.	37.6 s \pm 0.794 s
HanTa (hybrid) incl. lemm.	31.9 s \pm 1.26 s

4.4 Run Time

We measured the time needed to tag and/or lemmatize the first 1000 sentences from TüBa D/Z in a Jupyter Notebook on a Laptop with one Intel i7 2.7 GHz Processor and 8.0 GB RAM. The results are given in Table 7. Currently the results of the morphological analysis of each word is not stored. So after PoS Tagging of the whole sentence the words have to be analyzed again for lemmatization. Moreover only probabilities for each PoS and not the lemmata are stored in the model, so for lemmatization each word has to be analyzed, which is clearly reflected in the run time. We report results for tagging only (i.e. analyzing each word only once) and for tagging and lemmatization.

5 Discussion

Looking at the lemmatization, we see that our approach gives surprisingly good results: the approach in fact is quite naive, the morphological classes are too coarse-grained and the model is massively overgenerating and allowing for all kind of nonsense analyses. Nevertheless, in most cases the correct PoS and the correct lemma is predicted. On the Tiger corpus HanTa is even slightly better than the TreeTagger, on the TüBa D/Z Treebank the TreeTagger outperforms HanTa with half a percent. GermaLemma gives the best results on Tiger. However, GermaLemma uses a dictionary derived from the Tiger corpus, thus a comparison on these data is not fair. On TüBA D/Z GermaLemma does not perform very well, but this is due to the bad performance of the trigram PoS tagger that was used to provide GermaLemma with the PoS tags it needs. The results from Spacy in both experiments are much behind all other approaches.

HanTa’s accuracy on lemmatization (97.12 %) at first glance seems to be below the results of LEM-

MING reported by Müller et al. (2015) (98.10 %). However, these results cannot be compared directly. In the first place, the reported result is on one split from the Tiger corpus, but it is unclear, whether it is a contiguous or a random split. More important, we excluded all closed class words from the evaluation. Since most closed class words occur very frequently and are easy to lemmatize, including these words will improve the results.

In the evaluation of the PoS Tagger the first remarkable observation is the result from the TreeTagger that is noticeable below the evaluation results of Giesbrecht and Evert (2009). A possible source of difference could be the version of the Tiger corpus. Probably, Giesbrecht and Evert used version 1 of the Tiger corpus that consists of $0.7 \cdot 10^6$ tokens.

Here again the results from Spacy stay behind the other taggers. Interestingly, the baseline trigram tagger is almost as good as HanTa on the Tiger corpus, but on the Hamburg Dependency Treebank HanTa outperforms the baseline clearly. Thus, indeed, the careful splitting of a word into its stem and suffix has an advantage over just using the last letters of a word to guess its PoS.

6 Conclusion and future work

In this paper we have presented a simple approach to German lemmatization. We have evaluated the lemmatization on three different large corpora and shown that the results are close to results that can be obtained by state of the art tools and methods. Furthermore, we have shown, that the use of HanTa’s morphological analysis for unknown words in PoS tagging is more useful than using arbitrary length suffixes to guess the PoS. The PoS tagging using morphological analysis even outperforms other widely used PoS taggers.

In order to make HanTa a useful tool, we will work on the speed of the analysis, which is now clearly below that of most other tools evaluated here. Small improvements on the quality can be achieved by further development of the script generating the training data. Here e.g. a better treatment of adjectival nouns could help. Most interestingly, however, would be to see the effect of using more fine grained morpheme classes, including information on number, gender, tense, etc.

References

- Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories.*, volume 168.
- Thorsten Brants. 2000. Tnt: A statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing, ANLC '00*, pages 224–231, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Martin Braschler and Bärbel Ripplinger. 2004. How effective is stemming and decompounding for german text retrieval? *Information Retrieval*, 7(3):291–316, Sep.
- Grzegorz Chrupala, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with Morfette. In *LREC 2008*.
- Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):3.
- Peter Eisenberg. 1994. *Grundriß der deutschen Grammatik. 3., überarbeitete Auflage.* Metzler, Stuttgart/Weimar.
- ExplosionAI GmbH. 2019. German: Available pre-trained statistical models for german. <https://spacy.io/models/de>. Accessed: 2019-06-24.
- Kilian Foth, Arne Köhn, Niels Beuck, and Wolfgang Menzel. 2014. Because size does matter: The hamburg dependency treebank. In *Proceedings of the Language Resources and Evaluation Conference 2014 / European Language Resources Association (ELRA)*. Universität Hamburg.
- Alexander Gashkov and Maria Eltsova. 2018. Lemmatization with reversed dictionary and fuzzy sets. In *SHS Web of Conferences*, volume 55, page 04007. EDP Sciences.
- Andrea Gesmundo and Tanja Samardžić. 2012. Lemmatisation as a tagging task. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 368–372, Jeju Island, Korea, July. Association for Computational Linguistics.
- Eugenie Giesbrecht and Stefan Evert. 2009. Is part-of-speech tagging a solved task? an evaluation of pos taggers for the german web as corpus. In *Proceedings of the fifth Web as Corpus workshop*, pages 27–35.
- John A Goldsmith, Jackson L Lee, and Aris Xanthos. 2017. Computational learning of morphology. *Annual Review of Linguistics*, 3:85–106.
- Bart Jongejan and Hercules Dalianis. 2009. Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 145–153, Suntec, Singapore, August. Association for Computational Linguistics.
- Jakub Kanis and Luděk Müller. 2005. Automatic lemmatizer construction with focus on oov words lemmatization. In *International Conference on Text, Speech and Dialogue*, pages 132–139. Springer.
- Kimmo Kettunen, Tuomas Kunttu, and Kalervo Järvelin. 2005. To stem or lemmatize a highly inflectional language in a probabilistic ir environment? *Journal of Documentation*, 61(4):476–496.
- Markus Konrad. 2017. Lemmatization of german language text. <https://datascience.blog.wzb.eu/2017/05/19/lemmatization-of-german-language-text/>. Accessed: 2019-06-24.
- Kimmo Koskenniemi. 1983. *Two-level morphology: A general computational model for word-form recognition and production*, volume 11. University of Helsinki, Department of General Linguistics Helsinki.
- Barbara Lenz. 1993. Probleme der kategorisierung deutscher partizipien. *Zeitschrift für Sprachwissenschaft*, 12(1):39–76.
- Cristian Moral, Angélica de Antonio, Ricardo Imbert, and Jaime Ramírez. 2014. A survey of stemming algorithms in information retrieval. *Information Research: An International Electronic Journal*, 19(1):n1.
- Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274, Lisbon, Portugal, September. Association for Computational Linguistics.
- Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. SMOR: A German computational morphology covering derivation, composition and inflection. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May. European Language Resources Association (ELRA).
- H. Schmid, 1999. *Improvements in Part-of-Speech Tagging with an Application to German*, pages 13–25. Springer Netherlands, Dordrecht.
- Heike Stadler and Werner Wegstein. 2016. Encoding a derewo word-list. In *Varianz und Vielfalt interdisziplinär: Wörter und Strukturen*, pages 65–73. Institut für deutsche Sprache.

Heike Telljohann, Erhard Hinrichs, and Sandra Kübler.
2004. The tüba-d/z treebank: Annotating German
with a context-free backbone. In *Proceedings of
the Fourth International Conference on Language
Resources and Evaluation (LREC'04)*, Lisbon, Por-
tugal, May. European Language Resources Associa-
tion (ELRA).

The CLIPS (Computational Linguistics & Psy-
cholinguistics) research center. 2018. pattern.de.
[https://www.clips.uantwerpen.be/
pages/pattern-de](https://www.clips.uantwerpen.be/pages/pattern-de). Accessed: 2019-06-23.

To Act Or Not To Act - Annotating and Classifying Email Regarding Necessary Action

Veronika Hintzen

LMU Munich, Germany

v.hintzen@campus.lmu.de

Alexander Fraser

Center for Information and Language Processing

LMU Munich, Germany

fraser@cis.lmu.de

Abstract

Every user of email is aware of the problem of reacting to emails that require a time-sensitive action by the recipient while being overwhelmed by informational emails. We define a new classification problem to capture this distinction, creating comprehensive annotation guidelines and carrying out annotation. We carry out a proof-of-concept implementation of a classifier and discuss our future research which will result in a tool that is usable in an everyday business environment.

1 Introduction

The usage of email as a major communication tool has grown over the past 20 years. As per the current email statistics report by the Radicati Group, it was estimated that 3.8 billion users would receive 281.1 billion emails per day in 2018 with an estimated growth of about 4.4 percent each year (Radicati-Team, 2018). So a user receives on average about 74 emails per day. Carreras and Màrquez i Villodre (2001) discuss how users spend too much time sorting, with one problem being spam. But whereas spam filters nowadays work more and more efficiently and instant messenger services such as WhatsApp, Signal and Threema are on the rise for private communication - and thus keep the major load of non-work-related mail from our mailboxes, and for example Googlemail already provides a topic related sorting of the remaining emails to their users - many emails people receive at work still don't require immediate attention. In business, most emails still are basically only for information purposes, such as a report of a meeting or an invitation to a workshop. While these emails might be relevant and perhaps even time critical, there is no one waiting for the recipient's reaction to the email. One can assume that emails that contain a question or a task would need to be prioritized higher

than an invitation. Sorting through those emails and setting priorities by hand often takes up a lot of time and can be seen as a major distraction in a stressful work environment. Information emails get more attention than necessary, important emails get overlooked easily and the time that could be used for working on assigned tasks or even for breaks is diminished by the sheer amount of emails one has to manage.

While email providers nowadays allow users to create simple filters based on keywords, setting up these rules still takes up a lot of time and can be difficult (Gupta and Goyal, 2018). As was noted by Carreras (2001), most users waste a large amount of time in managing their emails or they prefer not to use keyword-based rules for filtering their email inbox. So, an automated tool that classifies emails regarding the expected attention that needs to be provided to them could help with prioritizing the received emails and thus improve the efficiency of work related communication.

Text classification in general and classification of emails in particular is a major subject in computational linguistics. Sebastiani (2001) defines it to be “the activity of labeling natural language texts with thematic categories from a predefined set” and considers it to be an instance of text mining, since ““text mining” is increasingly being used to denote all the tasks that, by analyzing large quantities of text and detecting usage patterns, try to extract probably useful (although only probably correct) information.” Thus, classifying emails regarding an action that is possibly expected from the recipient by the transmitter can be broken down into a bi-label or multi-label text classification problem depending on the desired degree to which the expected action should be distinguished. The general idea is to have a predefined set of labels or classes and find the class that best fits a given text. In this case, a binary label classification would simply be to sort the email into one of the two categories *ac-*

tion required and *no action required*, depending on whether there is any text in the email that indicates that the addressor of the email expects the recipient to become active in any way - for example to answer a question or to do a task. But it could also be interesting for the recipient to further discern between these two options and to have the emails labeled according to the degree of action that is required, as we will discuss later.

The paper is structured as follows. First a short overview over the annotation process will be given with an example email. Then the experiments done on these mails will be described and analyzed. This is followed by an outlook on possible improvements and a conclusion.

2 Annotation

In December 2001 the Enron corporation, one of the biggest energy companies of the US at the time, declared bankruptcy. In the ensuing investigation by the Federal Energy Regulatory Commission about 500000 emails by over 150 users were released to the public. It is perhaps the biggest publicly available email corpus and since then has been very popular with researchers. It contains a large variety of business emails as well as spam mails and private mails. A SQL-dump by Ruhe (2016) of the Enron data set was used. This is basically a “repaired” version of the MySQL-dump that was originally created by (Shetty and Adibi, 2004) but is no longer available. This MySQL-database contains all the info from the emails in a clean and easily retrievable format. For saving the annotations, the message-table was simply extended with the columns *label*, *notes* and *reviewed*.

Consider the email presented in table 1. The sender of this email obviously expects the recipients of this email to become active, which is implied by the following wording: “Can the two of you coordinate...” which would lead to this email being annotated with the *Action Required (AR)* category. But at the end of the email, the sender asks a question: “Can we get together that morning and review your analyses?” This means, the addressor expects a reply by the recipients, confirming this request for a meeting or maybe an alternative proposal. This leads to the annotation with *Reaction Required (RR)*.

Since the categories are considered to be hierarchical, a requested reaction is considered to be a little more important than an action, since the ad-

dressor might wait expectantly for the reply. Therefore, the possible *RR* annotation trumps the also possible *AR* annotation.

Also, as you can see, the email ends with “888-582-7421thankskh”, where obviously some whitespaces went missing, leading to the weird string “7421thankskh” being considered a token in this email.

1240 emails were randomly selected and sorted into one of eight categories. Table 2 shows them in their hierarchical order with their respective abbreviation, their count and a short description.

These labels were selected for their relevance regarding a work situation. While private emails might still have some of the same cues as business emails the annotators perceived their language and subjects as so strongly differing from the business emails that it was decided to create an own label for them. Some of these private mails for example contained jokes, cooking recipes or discussions about 9/11. For further information on the definition of these categories, the annotation guidelines are available at <http://hintzenv.wordpress.com>.

While non-relevant emails were usually easy to annotate, sometimes missing context made it hard to decide on a label. These emails were annotated “Unsure” and reviewed again later. Those that still would not be clear stayed in that category. Also, the announcement of a birthday cake at a colleagues cubicle led to some discussion. It was decided it would be considered to be an invitation.

3 Experiments

3.1 Evaluation Techniques

To evaluate a classifier, after the classification of the test data, the appointed classes need to be compared with the originally annotated classes. There are several common practices to evaluate the performance of a classifier. Precision, recall and f-score were computed for each class used by each of the implemented classifiers. Confusion matrices are also presented.

3.1.1 Micro and Macro scores

For a general overview of the different models, the micro and macro averaged scores will be computed, which show a weighted (micro) and unweighted (macro) average score of the performance of a model.

The procedure for computing these scores is

subject	body
SFV Rate Design for Sun Devil	<p>James &David -\t</p> <p>Can the two of you coordinate a revenue model for Sun Devil that incorporates a Straight - Fixed Variable rate design along the following parameters using 15, 20 and 25 year terms:</p> <p>San Juan utilization: 85% of 780,000/MMBtu/d</p> <p>Mainline: 85% utilization of 810,000/MMBtu/d</p> <p>Phoenix lateral: 75% utilization of 500,000 MMBtu/d</p> <p>Also, prepare some ROE sensitivities if the above utilization falls by 10% and rises by 10%.</p> <p>I will be out of town till Tues 11/13. Can we get together that morning and review your analyses?page me if you have questions at 888-582-7421thankskh”</p>

Table 1: Example email from the Enron data set

category name	label	count	description
Reaction Required	RR	259	A reaction to the email is required.
Action Required	AR	87	The recipient is required to take an action.
Appointment/Deadline	AD	94	The email contains an appointment or deadline.
Invitation	I	25	The email contains an invitation.
Contains Information	CI	518	The email contains business-relevant information.
Private	P	135	The email is private, not business relevant.
Non-Relevant	NR	113	The email is business-related, but not relevant (e.g., newsletters).
Unsure	U	9	This is a catch-all category, see the discussion in the text.

Table 2: Categories with respective counts

described by Yang (1999), and Tsoumakas et al. (2010) present the respective formulas. Furthermore, for the micro average score, Asch (2013) shows, that for single-label classifiers the scores for precision and recall are equal. Since the F-Score is the harmonic average between Precision and Recall, and for the micro-average-score, those two scores are equal, so is the F-Score. In this paper, the macro averaged F1-Scores (later in this paper referred to as *macro score*) will be compared with the micro averaged F1-Scores (later in this paper referred to as *micro score*).

3.2 Preprocessing

In order to get the emails into a processable format, the bodies have to be tokenized: special characters, punctuation, tabs and newlines were filtered out and the text was split on blanks. We use Word2Vec word embeddings and create a single average vector for each document. We split the annotated data into 80 percent training data and 20 percent test data.

Because the annotated data has been labeled in a way that is quite fine grained and the counts vary

greatly between the categories, the classifiers were tested on different groupings of the labels, which will be described in detail in the evaluation section. These groupings were selected by their intuitive relevance to everyday working life and in the hopes of finding a grouping that gives a balanced overall performance.

3.2.1 Word Embeddings

For this study, Word2Vec word embeddings were used. Word2Vec models with the dimensionalities of 50, 100, 200 and 300 were trained on the Enron data set and thus on about 62 Million tokens - and a vocabulary of about 650 thousand unique tokens. We do not use pre-trained Word2Vec embeddings because the Enron data set consists of emails, and emails are of a different nature than, for example, Wikipedia articles regarding the used vocabulary, syntax and the existence of many typos. The idea was to use Word2Vec embeddings trained on the Enron data set in order to better account for these errors and inconsistencies. We also tried pre-trained GloVe embeddings in initial experimen-

	micro	macro
Naïve Bayes	0.583	0.426
SVC baseline	0.538	0.321
Word2Vec 50	0.551	0.501
Word2Vec 100	0.571	0.489
Word2Vec 200	0.543	0.473
Word2Vec 300	0.575	0.499

Table 3: Micro and macro scores of all models on non-grouped classes

tation, but the results were much worse, and so we did not continue experimentation with them. We leave further study of this issue for future work.

3.3 Classification

Afterwards the classifier was trained on the vectorized texts with their respective labels from the training sets and with the learned features the vectorized texts from the test set are classified. In our study, six different classifiers (Naïve Bayes, baseline SVM on words, four SVMs for the different dimensionalities of Word2Vec embeddings) were combined with different groupings of the labels.

3.4 Evaluation

In order to evaluate the performances of the different classifiers first an overview of the micro and macro overall scores will be shown. For a more detailed look into the models, the Precision, Recall and F1-Scores of the classes in the best- and worst-performing models will be presented.

3.4.1 Overview

A first test with separate categories produced very unsatisfactory results. Table 3 shows the weighted and unweighted average F1-Scores for each of the models. For purposes of readability the scores have been rounded to the third decimal place.

According to the (weighted) micro-Score, the Naïve Bayes classifier performs best at this task, but a look on the unweighted score shows that the small classes are classified significantly worse than the larger classes. With the (unweighted) macro-Score, the model that resulted in the highest score in our tests would be the Support Vector Classifier based on the 100-dimensional Word2Vec-embeddings.

Overall, these scores are not really satisfactory. This is not surprising due to the small size of training data per class. In the test set, the smallest class

Name	Category grouping
7-base	RR, AR, AD, I, CI, P, NR
2-action	RR+AR, AD+I+CI+P+NR
2-timecrit	RR+AR+AD, I+CI+P+NR

Table 4: Keywords assigned to groupings

only had seven occurrences. So, the idea arose to group the categories in order to get more training and testing examples for each class.

A more detailed look into the performances of the models with the base task will follow in section 3.4.2, where the best- and worst-performing models will be discussed in detail.

For an overview of the performances, tables 5 and 6 compare the micro and macro scores for each grouping in each of the models. The weighted micro scores take into account the size of the groups. The unweighted macro-scores do not do that. While usually one would think that the weighted performance of a model would give more insight into the performance of a model, we decided to add the unweighted scores since the category containing *CI* shows the highest F1-Scores due to the size of the corresponding data set but is one of the lesser important categories, and as such the weighted scores tend to skew the performances in favor of the bigger and less important categories.

We now discuss two further groupings of the labels we experimented with. Category groupings are assigned a name, leading with the number of classes the grouping results in, followed by a short keyword for the way criteria they are grouped for. In table 4 you can see these names with their respective assigned grouping.

For reasons of readability, again the scores were rounded to the third decimal places and the model names have been abbreviated: *nb* for *Naïve Bayes*, *svc bl* for *baseline Support Vector Classifier*, *wv* for the SVC using the self-trained Word2Vec embeddings. Also, for reasons of clarity, the table 5 will refer to the micro-scores (weighted), while table 6 will refer to the macro-scores (unweighted). For each grouping the best weighted and unweighted scores are underlined. The highest weighted and unweighted F1-Scores across all models are shown in **bold**.

As one would expect, the best performing groupings are those that contain only two classes and the grouping with each label on its own performs the worst. Also, as expected, the unweighted

	7-base	2-action	2-timecrit
nb	0.583	0.725	0.656
svc bl	0.538	0.757	0.725
wv50	0.551	0.729	0.676
wv100	0.571	0.696	0.709
wv200	0.543	0.721	0.692
wv300	0.575	0.741	0.700

Table 5: Comparison of micro-Scores

	7-base	2-action	2-timecrit
nb	0.426	0.574	0.616
svc bl	0.321	0.612	0.641
wv50	<u>0.501</u>	0.611	0.630
wv100	0.489	0.580	<u>0.667</u>
wv200	0.473	0.622	0.646
wv300	0.499	<u>0.652</u>	0.658

Table 6: Comparison of macro-Scores

scores are almost consistently lower than the micro scores while the pre-trained embeddings have an almost consistently worse average F1-Score than the Word2Vec embeddings. While the differences especially in those scores that are very close to each other cannot be considered statistically significant, this paper only strives to discuss the possibility of the task and possibly provide scores for comparison with similar future tasks.

It is noteworthy that, when comparing the micro scores for one grouping, the scores are surprisingly uniform with at best a difference of 0.081 between the worst and best performing models and even only 0.061 in the 2-*action*-grouping. This can be attributed to the class sizes. In the micro score the larger a class the higher the influence on the resulting average score. A look on the respective precision and recall scores of the classes shows consistently good performance on these larger classes in all the models.

In order to see how a model improves when being trained on fewer classes with more training examples, you can compare horizontally and see a mostly consistent increase in performance from seven classes to two classes.

When looking at the two binary groupings, with micro averaged scores, it seems as though the switch of the *AD*-class from the larger class to the smaller group actually decreased the overall performance. But in table 6, you can see that there,

			p	r	f1
svc bl	AD+I+CI+P+NR	0.77	0.95	0.85	
	RR+AR	0.67	0.26	0.38	
w2v100	I+CI+P+NR	0.77	0.80	0.78	
	RR+AR+AD	0.57	0.53	0.55	

Table 7: Scores of the best-performing models

	Actual						
RR	RR	AR	AD	I	CI	P	NR
RR	16	4	2	-	15	1	2
AR	1	2	-	-	5	-	-
AD	2	-	7	-	4	-	-
I	-	-	1	4	1	-	-
CI	29	12	4	3	80	4	4
P	3	-	-	-	3	13	5
NR	-	-	-	-	5	1	14
Total	51	18	14	7	113	19	25

Table 8: Confusion matrix for Word2Vec50 based SVM on base task 7-*each*

too, is an improvement. This leads to the conclusion that the performance of the larger class drops, while the performance of the smaller class - which would be considered more important in a business environment - improves.

When looking at the micro averaged scores, the comparison of higher dimensionalities of the Word2Vec embeddings with the baseline SVC also seems notable. While with more classes the higher dimensionalities seem to add to the performance, with the binary classifiers, the higher dimensionalities perform even worse than the baseline.

With the macro averaged scores, this effect vanishes and the Word2Vec-models perform consistently better than the baseline model, again indicating that the baseline SVM has a bias towards larger classes.

3.4.2 Detailed discussion of best- and worst-performing models

For a detailed analysis of the best- and worst-performing models, table 7 shows the precision, recall and f-scores for each class. For the model performing best regarding a weighted calculation of the average value - the SVC baseline model with the 2-*action*-grouping (5:2) - , the recall is very high for the larger group containing the less important categories while the recall with the important categories is very low with only about 0.26. Using

the best model by macro score, you have a lot less variability within the scores.

The worst-performing model with both, the micro scores as well as the macro scores, was the SVM with the 50-dimensional GloVe-embeddings in the “7-base” task - where there were no groups, but each label for its own. In fact the labels *RR*, *AR* and *I* were not classified at all - with 0.00000-scores, resulting in the low macro scores. We do not present results on pre-trained GloVe embeddings in detail, leaving a study of how to adapt pre-trained embeddings to the Enron corpus for future work.

3.4.3 Evaluation and Error Analysis - Word2Vec50 with the 7-each grouping

With the 7-each “grouping” being the base task of this project, the best performing model of this task will be discussed here.

In order to get a better look on the distribution of the actual and predicted classes, in the following the confusion matrix for the 50-dimensional word2vec model (see table 8) will be presented.

3.4.4 Evaluation and Error Analysis - SVC baseline model with 2-action

In order to get more detail on the performance of the model, and to get an idea of where the performance issues arise from, a detailed confusion matrix for the svc baseline model with the category 2-action grouping is presented in table 9.

For readability the larger group (*AD+I+CI+P+NR*) will be shortened to *Other* and for reasons of space-usage, the confusion matrix will have the actual categories on the X-axis and the predicted categories on the Y-axis.

While the performance here is significantly better than with the GloVe-models and at least $\frac{2}{3}$ of the mails labeled with *Action* also really require said action, still 51 of the 69 mails will be lost - that’s almost $\frac{3}{4}$.

If you look at the distribution of the *RR* and *AR* emails you can see that with 0.28 the share of correctly classified emails in the *AR* category is only slightly bigger than the 0.26 in the *RR* category. But if you consider that the *AR* category is significantly smaller than the *RR* category, with only one additional misclassified email, that percentage would have dropped to 0.22. So, it is safe to assume that both labels are classified with a comparable performance.

What is interesting to see, though, is, that within

	Actual						
	RR	AR	AD	I	CI	P	NR
AR+RR	13	5	-	-	6	3	-
Other	38	13	13	7	108	16	25
Total	51	18	13	7	114	19	25

Table 9: Confusion matrix for svc baseline model with 2-action

	RR	AR	AD	I	CI	P	NR
AR+RR	23	6	2	-	21	-	1
Other	28	12	11	7	93	19	24
Total	51	18	13	7	114	19	25

Table 10: Confusion matrix for Word2Vec300 model with 2-action

the group of emails that were wrongly classified as *AR + RR*, these emails originally stem only from the *CI* and *P* categories. All other categories were classified correctly. But here with three of the nine wrongly classified emails being *P* category emails, the misclassification of *P* emails is significantly higher (0.16) than of the *CI* emails (0.05). This might be traced to the fact that often private emails also contain requests for an action or a reaction.

For comparison, consider the confusion matrix of the Word2Vec300 model trained with this grouping that performed second best to the baseline svc - best with the macro Score, see table 10.

This confusion matrix produces a very different picture than the baseline svc. Here already $23 + 6 = 29$ of the $51 + 18 = 69$ actual action requiring emails are found - which is already over 40 % -, there are a lot more misclassifications towards the smaller class.

3.4.5 Evaluation and Error Analysis - Word2Vec100 model with the 2-timecrit grouping

In order to again get a better look at the performance of the Word2Vec100 model in the 2-timecrit grouping, see table 11.

With this SVC and grouping - while here, too, there are a lot more misclassifications toward the smaller class, instead of $\frac{3}{4}$ of the action requiring mails being “lost”, of the total 82 mails regarded as *ActionRequiring*, only 38 are missed. With that being less than the half, that’s already a lot less than with the baseline svc in grouping 5.

It is interesting to see that with the *AD* emails there is an unusually high recall with 10 of 13 being classified into the action requiring group.

For comparison, here, too, shall be presented the confusion matrix for the baseline svc model which performed second-best in the *2-timecrit* grouping (table 12).

Here, again, the *AD* category - while having only eight of the thirteen emails classified correctly - performs surprisingly well. Again, the misclassification counts towards the smaller class are smaller, while those towards the larger class are stronger.

This leads to the conclusion that overall the vectors produced by the CountVectorizer lead to a tendency of classifying in support of the larger class and the vectors produced, while the average document vectors resulting from the Word2Vec-embeddings lead to a tendency of classifying in support of the smaller class.

In an everyday office life the latter would probably be preferable. Consider an email account containing different folders for each class and the user - running from one appointment to another - only wanting to see the action-relevant emails when looking into the respective folder. While having emails there that don't belong would be considered a nuisance, missing emails might prove to be a problem of a lot bigger scale.

4 Outlook

In this section, an outlook on possible improvements that can be made on and with the existing models, as well as ideas for future work - i.e. possible variations of the tasks - will follow.

While none of the models implemented yet proved to be adequate for an everyday use in a work environment, there are several possibilities to improve the performance and ideas that might prove to be worth looking into.

A larger annotated data set should help greatly in the training of the used models. Language is too complex to grasp meanings just from a little over a thousand emails. With larger annotated data sets, more features can be accounted for and so the non-binary models' performance might also improve. Possible ways to achieve a larger data set include: *more time, more personnel* and using *distant supervision*. Although the annotation with distant supervision produces annotations of a relatively bad quality, it might still be better than only working with a small data set. In contrast to that, the usage of more personnel would help in ensuring a high quality of annotation. So a good compromise between the two could possibly be found, where

	RR	AR	AD	I	CI	P	NR
AR+RR+AD	26	8	10	3	24	2	4
Other	25	10	3	4	90	17	21
Total	51	18	13	7	114	19	25

Table 11: Confusion matrix for Word2Vec100 model with *2-timecrit* grouping

	RR	AR	AD	I	CI	P	NR
AR+RR+AD	18	3	8	-	12	3	1
Other	33	15	5	7	102	16	24
Total	51	18	13	7	114	19	25

Table 12: Confusion matrix for svc baseline model with *2-timecrit* grouping

a semi-large set of high quality annotations would be combined with a large set of low quality data and where good results might be achieved by having several training iterations with only the first iteration on the combined data set and the other iterations on only the high quality data set.

Another possibility to get larger annotated data sets would be to use active learning and let users help with annotating emails in a run-time environment and therefore improve the used classifier according to the user's needs. This would also have the advantage of having the models trained on more contemporary business emails being adapted regarding the respective business area of the user and to the change of their vocabulary in the past years.

With larger data sets the application of deep learning models would become possible. With the *Support Vector Machines* much information is lost during the reduction of the embeddings to an average document vector, so a model that is able to properly grasp the multiple dimensions of the embeddings could possibly find more and better relations between the features and the corresponding classes of the documents and thus make better classifying decisions.

In addition, one possibility to improve the performance could be found by including the subject lines as well as info about the sender and the recipient/s of the emails into the models since these already provided relevant information about the context of the mail during the annotation process.

In emails you often have the former emails from the exchange appended to the latest email. While in the annotation process these old emails sometimes provided necessary information on the context of the email, in a bag-of-words model, that is used in

Naïve Bayes as well as in Support Vector Machines, would give these old mails too much weight. Especially in long email exchanges, with five and more emails and possibly only a short question in the latest mail. A weighting of the words depending on their occurrence location might prove to be useful.

The Enron data set has often been used for training classifiers for spam mail. Instead for this project the obvious spam mail has simply been categorized as Non-Relevant. The implementation of a spam filter based on a larger spam-specific training set that is run before our classifier to eliminate obvious spam might also improve the results (and/or help with future annotation of new data).

Several of the produced errors might have their origin in the hierarchical nature of the categories. This problem might be evaded by allowing the annotation with more than one category per email and/or by using classifiers that produce more than one label per email and then tweaking these classifiers by weighing those categories in favor of the action-inducing categories.

5 Conclusion

For this paper, a classification task was set up from scratch. The goal was to build a classifier that could distinguish between emails regarding whether a response or other action was required from the recipient. Without suitable annotated data being accessible and possibly even existent, first, a data set had to be annotated by hand.

Being probably the biggest open source data set for business emails, the Enron data set was selected as a foundation for this self-annotated data set. The annotation produced an annotated data set of 1240 emails. Due to the nature of the Enron data set, the distribution of the categories was rather imbalanced leading to very different sizes of learning and test sets for each of the categories.

For the task of building a classifier, it was decided to compare six different models: one Naïve Bayes classifier, a baseline Support Vector Classifier as well as four Support Vector Classifiers based on Word2Vec embeddings of different dimensionalities.

Since the performance of these classifiers on the base task with a class for each of the categories was not satisfying, the categories were grouped in different ways with the goal of finding a grouping that would perform better and still be of practical use in an everyday work environment.

As was to be expected, the two groupings leading to binary classifiers performed far better than the multi-class classifier. With a binary classifier, the training and test data sets were both bigger and much more balanced. Also, with a binary classifier, there are far less classes that can be taken for misclassification. Also, it was interesting to see, how the performance of the embeddings-based models changed with additional dimensions. While the Word2Vec embeddings would produce varying micro and macro scores, we noticed that the pre-trained GloVe embeddings - while over all having a worse performance than the Word2Vec based models - showed constant improvement of performance with additional dimensionalities on each of the tasks (but we omit these detailed results). Another interesting result was that the two baseline models had a surprisingly good performance overall.

In the detailed error analysis of the best- and worst-performing models, it was implicated that additionally to the small size of the data set, the hierarchical order of the categories might have been one of the major origins of misclassifications, since this led to fewer distinguishable features of said categories. The most improvement could possibly be achieved by improving the used data set regarding its size and quality. But also with the used classifying models, there are many possible tweaks and changes that could be tested and that might prove to have quite an impact on the classifying performance.

In conclusion - while at least on the binary tasks promising results could be achieved - none of the presented models has a performance that would be good enough for practical use. Too many misclassifications would make a tool based on the models used here very frustrating to work with. Also, this would probably even lead to financial risks when an email that requires a time-sensitive action by the recipient, would not be recognized as such by the classifier. But even with these not yet satisfactory results, it was shown, that this task is not impossible to achieve but rather a question of obtaining bigger data sets. Using the annotation guidelines and initial data set that we have created in this work (and make available with the publication of this paper), it will be possible for interested researchers with access to more resources to create a much larger training corpus than we were able to create. In addition, we plan to study how to incorporate

active learning to learn from the user as they identify mis-categorizations as an additional way to obtain further supervision for this important task, see, e.g., the work of Tong and Koller (2002), as well as more recent work. Finally, once we have further supervision available for this task, we will study (data-hungry) classification models based on neural networks, from which we expect to obtain further improvements in performance.

Acknowledgments

The hardware that was used for this research was provided by the Steering Lab at Horvàth & Partners Munich.

References

Vincent Van Asch. 2013. Macro-and micro-averaged evaluation measures [[basic draft]].

Xavier Carreras and Lluís Márquez i Villodre. 2001. Boosting trees for anti-spam email filtering. *CoRR*, cs.CL/0109015.

Deepak Kumar Gupta and Shruti Goyal. 2018. Email classification into relevant category using neural networks. *CoRR*, abs/1802.03971.

Radicati-Team. 2018. Email statistics report, 2018–2022. Technical report, The Radicati Group, Inc., March.

Arne Hendrik Ruhe. 2016. <http://www.ahschulz.de/enron-email-data/>.

Fabrizio Sebastiani. 2001. Machine learning in automated text categorization. *CoRR*, cs.IR/0110053.

Jitesh Shetty and Jafar Adibi. 2004. The Enron email dataset database schema and brief statistical report. Technical report.

Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, March.

Grigoris Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2010. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–685.

Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1):69–90, Apr.

Akku•Bohr•Hammer vs. Akku•Bohrhammer: Experiments towards the Evaluation of Compound Splitting Tools for General Language and Specific Domains

**Anna Häfty^{1,3}, Ulrich Heid², Anna Moskvina²,
Julia Bettinger^{1,3}, Michael Dorna¹ and Sabine Schulte im Walde³**

¹Robert Bosch GmbH, Corporate Research

² Institute for Information Science and Natural Language Processing,
University of Hildesheim

³Institute for Natural Language Processing, University of Stuttgart
{anna.haetty, michael.dorna}@de.bosch.com,
{heidul, moskvina}@uni-hildesheim.de,
{julia.bettinger, schulte}@ims.uni-stuttgart.de}

Abstract

We present a comparative evaluation study for splitting German compounds which belong to general language or to a specific domain. For the domain, we focus on DIY (“do-it-yourself”). The study consists of two parts: First, we evaluate three tools for compound splitting in German, one based on lexicons and corpus frequencies and two based on language-independent statistical processing. We introduce the tools, discuss the data and the construction of a gold standard, and show first results for binary and ternary noun compounds, as well as for the handling of non-splittable items. In a second experiment, we post-train one of the splitters with text data from the DIY-domain, and evaluate the splitting performance on domain-specific compounds.

1 Introduction

German is a highly compounding language, which means that several simple words like *Akku* “battery”, *bohren* “to drill” and *Hammer* “hammer” are combined to form a complex word like *Akkubohrhammer* “cordless hammer drill”. As a result, these complex compounds can be rather infrequent. In order to automatically process them, it is often useful to split them into their (usually more frequent) components, by using a compound splitter. However, compound splitting is a complex task, because there are often several splitting options possible. Splitting compounds which originate from specific domains further aggravates the problem: Both compounds and components might be even more infrequent, and a splitter might not

have seen such data in the training stage, because it was trained on general language data.

For those reasons, we establish two evaluation settings to get a better insight into compound splitting for general language and for specific domains: (i) we compare several splitters with respect to their performance on both general language and domain-specific compounds and (ii) we post-train a splitter with domain data and evaluate the effect on domain-specific compounds.

In the first setting, we report on the comparative evaluation of three published tools. As a basis we use data from a specialized corpus, a general language corpus and the word formation literature. As the application domain is do-it-yourself instructions (DIY) from online forums, and we targeted the extraction and semi-automatic description of terminology candidates from the forum texts, compound splitting was mainly addressed with ontology building in mind; typically, heads of determinative noun compounds are hypernyms of such compounds. By splitting a noun like *Bandssäge* (“bandsaw”) into *Band•säge*, the noun *Säge* can be identified as a hypernym of *Bandsäge*. Consequently, we only worked on noun compounds so far, even though adjective compounds would be equally interesting and even less covered by state of the art analyses of compound splitting. While split points are the main issue when it comes to the quality of the analysis of binary compounds, structure plays a major role for ternary compounds and items composed of more than three morphemes. Thus, for tri-morphemic compounds, we assessed both morpheme decomposition and structure assignment.

In the second setting, we post-train one of the compound splitters on a DIY text corpus. We then

split all noun compounds in the corpus using the original and the modified splitter, and compare the results.

The paper is organized as following: In section 2 we will give an overview about the related work and in section 3, we introduce the three compound splitters. Section 4 describes the data that were used for the first experiment; additionally it gives details about how to create the compound gold standard, and how it can be used for evaluation. Section 5 describes the settings of the second experiment, how to post-train a splitter and which data were used. In section 6, we perform a detailed evaluation of the experiments. In section 7, we present and discuss aspects of the outcome of our evaluation, and in section 8, we conclude and point to needs with regard to future actions.

2 Related work

There exist a variety of compound splitters, which rely on different methodologies. There are linguistically motivated splitters, that rely on word frequencies (Koehn and Knight, 2003; Cap, 2014; Weller-Di Marco, 2017). CharSplit (Tuggener, 2016) however relies on a character-based method. A recent trend is to exploit distributional semantics to find the correct components (Ziering et al., 2016; Riedl and Biemann, 2016). Similarly, another splitter relies on semantic analogies (Daiber et al., 2015). Beside using different methodologies, the splitters return different splittings. For example, the Simple Compound Splitter by Weller-di Marco (2017) can return a binary or an n-ary split, lemmatize and POS-tag the components. CharSplit, however, does only a binary splitting. The output might depend on the application the splitter was designed for; for example, CharSplit was designed to find the compound heads in order to facilitate coreference resolution.

To our knowledge, no huge compound splitter comparison exists; Escartín (2014) conducts a small comparative study with two compound splitters. In addition, there is little work on domain adaptation of compound splitters. Macken and Tezcan (2018) perform Dutch compound splitting, and adapt the splitter to the automotive and the medical domain. They find that only using general language data performs better than only using domain-specific data, but a combination of both leads to the best results.

3 Tools for splitting German compounds and their evaluation

While a number of well-known and some upcoming tools for splitting German compounds exist, we are not aware of recent activities towards the comparative evaluation of the output quality of such tools. An older landmark for word formation evaluation of German as a whole is the Morpholympics contest, held in 1994 (Hauser, 1994). We briefly report about both, tools and evaluation.

3.1 Tools for compound splitting

In a general way, and especially with a view to the kind of evaluation we carried out, tools for compound analysis may be subclassified according to the kind of output they provide:

- tools only providing morpheme decomposition;
- tools providing morpheme decomposition and one or more structure proposals.

In addition, one may consider further types of tool output, e.g. category values of the morphemes identified. While this classification is based on the kinds of output produced by the tools, one may also distinguish symbolic vs. hybrid vs. purely statistical, machine learning based tools, according to the approach. In the following, we briefly describe the tools we analyzed, and we mention a few more that may be used in a second round of the evaluation.

3.2 SECOS: Unsupervised Compound Splitting With Distributional Semantics

Unlike most systems that rely on dictionaries or are trained in a supervised fashion, SECOS (Riedl and Biemann, 2016) relies entirely on distributional semantics. The hypothesis investigated by the researchers postulates that compounds are similar to their constituting word units. Their method is based on a distributional thesaurus that is computed using a tokenized monolingual background corpus without any additional linguistic processing. The first step is the extraction of a candidate word list that defines the possible word units of compounds. The second step is splitting the compounds. The last step is a ranking of the splits and returning the top-ranked ones. The method is proven to be language independent: several experiments were conducted on German and on Dutch, they produced equally good results. The tool is freely available.¹

¹<https://github.com/riedlma/SECOS>.

3.3 Compound splitting tool from Tübingen University

The authors (Ma et al., 2016) introduced a letter sequence labelling approach, which can utilize rich word form features to build discriminative learning models that are optimized for splitting. The prediction of labels is achieved by training conditional random fields. The method is language-independent and does not require any linguistical preprocessing. Splitting is conducted at the surface form level. The current system, available for testing, is trained to split multi-constituent compounds at the boundaries of all the constituent words, instead of only splitting at the top level (complete morpheme decomposition).

3.4 CompoST: Compound Splitting Tool

The tool splits compounds into their morphemes using morphological rules and corpus frequencies. The underlying method (Cap, 2014) involves using the geometric mean of subword frequencies to disambiguate possible splits. CompoST was developed for compound processing in statistical machine translation, but it can equally be used as an independent module for morphological analysis. It requires frequency counts derived from a corpus; candidate items are analysed by SMOR (a rule based morphological analyser for German) (Schmid et al., 2004). CompoST allows to set different parameters and therefore to gain different versions of output. For instance, it can split a word even when frequency scores suggest that the word can not or should not be split (forced splitting), or it can split only nouns. One of the drawbacks of the tool is that words unknown to SMOR cannot be split, as well as disambiguation of possible splits is entirely based on frequency, and this might lead to inconsistencies on a non-lemmatized word list.

4 Gold standard for compound splitting

A gold standard evaluation was carried out, in the framework of our project on term candidate extraction from do-it-yourself instructions (DIY). While the focus of the evaluation was on the coverage of the data from the DIY-corpus, and on the quality of the respective analyses, we also wanted to explore the performance of the tools on general language data. We created a database that contains the gold standard, as well as the output of individual tools. In this way, all elements of the evaluation can later be enhanced: more gold data can be added, and the

results of further tools can be compared.

4.1 Sources and selection criteria

For both, specialized and general language, corpus data were used, but with different objectives. For specialized language, we used a corpus of 11 million running words, composed of expert texts and user generated content (=UGC) from the domain of DIY instructions. The relationship between expert and UGC texts was roughly 1:5. For the gold standard, we extracted noun compounds (by means of TreeTagger-assigned pos=“NN” annotations) from three frequency bands: top, medium and low frequency items. Given the overall frequency distribution of nouns, the distribution of candidate items shown in Table 1 was achieved.

We are aware that the “medium” frequency band is as yet underpopulated. Additional sampling may be needed to provide roughly the same quantities of data as for the two other frequency bands. However, this would not even out the relationship between binary and trimorphemic candidates, which is uneven as well but likely relatively close to the distribution to be expected in the texts under analysis. To counterbalance the almost proportional sampling from the specialized corpus, we added data from general language materials. In this part of the gold standard, we did not aim at replicating frequency distributions from a given corpus, but we rather targeted a collection of all cases that are discussed as relevant in the literature on German compounding. This approach is similar to part of Hauser’s (1994) sampling method. Thus ca. 200 items were taken from the standard handbook on German morphology by Fleischer and Barz (1995). We cross-checked however the chosen items against 200 M words of news texts and against the SdWaC corpus (Faaß and Eckart, 2013), and only used items present in at least one of them. These items provide a wide range of possible issues for compound splitting, e.g. adjectival non-heads that are not in the positive form (*Mehrarbeit* “additional work”; *Reinststoff* “ultrapure substances”, lit.: “ultrapurest substances”) or compounds with phrasal non-heads (*Heißwasserspeicher* “boiler”, lit.: “hotwater storage”).

4.2 Annotation of the gold standard

The annotation was carried out manually, by one linguist. The reason why we consider this sufficient is that the underlying guidelines are based on standard analyses from morphological theory

frequency range	frequency	non-split	binary	trimorph.	total
top	$f > 100$	44	329	67	440
medium	$41 > f > 37$	6	113	29	148
low	$f=12$	21	312	100	433
total		71	754	196	1,021

Table 1: Frequency-based sampling of noun compounds from an 11 M word corpus of DIY forum texts.

(Ortner et al., 1991; Pümpel-Mader et al., 1992; Fleischer and Barz, 1995; Donalies, 2011; Donalies, 2014); for items which, according to these sources, can receive more than one analysis, all valid analyses were included in the gold standard, such that tools providing one of them were not punished. The annotated data were stored in a database. The following features were annotated:

- split points on the form level - in the sense of Koehn and Knight (2003) - and lemma forms of the morphemes;
- pos categories of the non-head morphemes;
- structure of tri-morphemic compounds (left vs. right branching).

In addition, the following documentary data were annotated by automatic means:

- number of split points (for easy counting of over- and undersplitting cases);
- lemma frequency of the item tested, as well as of its components in 200 M words of news text and in SdWaC.

The following is a simplified example of the linguistic representation of the items in the gold standard database; the first feature is the POS combination of the non-head morphemes; it is followed by the lemma from the corpus, its decomposition into morphemes at the level of surface forms, its topmost split at the level of surface forms, as well as the morpheme decomposition and the structure proposal (=topmost split) on the level of lemmas.

```
adj-v Kleinstlebewesen
  - kleinst lebe wesen + kleinst
    Lebewesen
  - klein leben Wesen + klein
    Lebewesen
```

The double annotation, at both lemma and surface level, ensures compatibility with most types of tool outputs and thus eases the comparison.

4.3 Data annotated

As mentioned above, we included noun compounds of three kinds in the database: binary and tri-morphemic compounds, but also items that cannot be split, e.g. because they are derivation products. We also included ca. 30 items which allow for two structural analyses, e.g. *Meerwasserentsalzungs•Anlage* vs. *Meerwasser•Entsalzungsanlage* (“desalination plant”, lit.: “sea water desalination plant”). The distribution over the full data set is given in table 2.

frequency range	#
non-splittable	86
binary	
- N+N, Adj+N	715
- V+N	118
tri-morphemic	294
total	1,239

Table 2: Distribution of compounds over the full data set.

5 Post-training with domain-specific text data

Adapting a compound splitter to a certain domain of interest, as DIY in our case, might improve the compound splitting for two reasons: First, the domain-specific components of a compound might be infrequent in general language, and that is why the correct split or base form of the component cannot be found. For example, the compound *Eloxierverfahren* (“anodizing procedure”) should be splitted and lemmatized to *eloxieren•Verfahren* (“to anodize•procedure”). Secondly, splitting probabilities might be skewed because a certain split is more likely in general language, while another one is more likely within the domain. For example, the compound *Rohrverbinder* (“pipe connector”) is likely to be split as *Rohr•Verb•Inder* (“pipe•verb•Indian”) in general language, because the three components do occur more often in gen-

eral language than the correct components *Rohr* (“pipe”) and *Verbinder* (“connector”).

However, post-training of a compound splitter on a domain-specific corpus is not always possible. It depends on the design of the tool and if the original training data are available for updating.

We adapt the splitter CompoST. CompoST relies on frequency counts derived from a corpus, in the default case a general-language corpus. To adapt the splitter to the DIY domain, we compute all the frequency counts for a DIY text corpus. Then we either add the frequencies to existing token entries, or create new ones. We use a domain-specific DIY corpus with 5.6 million words. The texts were collected from different sources, but all of them are DIY-related. There are texts produced by domain experts as well as by interested lay users, such as encyclopedia texts, DIY-instructions and manuals. Preprocessing has been done with SpaCy² (Honnibal and Johnson, 2015). Working with the German language model of SpaCy, we make use of the tokenizer, the POS-tagger and the lemmatizer. While the tagging itself is based on a convolutional neural network, the lemmatizer still works with a conservative look-up table. We use the POS-tags to select noun compounds as candidates for compound splitting.

6 Evaluation

6.1 Comparison of compound splitters

6.1.1 Evaluation methods

We mainly follow Koehn and Knight’s (2003) procedures for the comparison of our gold standard splits with the output produced by the tools. To ease the quantitative assessment of over- and undersplitting, we count the number of split points in each gold standard item and in each tool output for the respective item and annotate this number back into the database. As we offer the gold analyses both on word forms and on lemmata, we use both versions as alternatives to match the tool output against: the results of each tool (or of each version of tool output) are inserted, for each gold item, into the respective row of the database table; for each tool output, the table is thus enlarged by one or several complete column(s). Not all tools provide just the split points; some provide in addition pos-features or other descriptive output. When preprocessing the tool output we keep track of such

specificities. We evaluated the analyses provided by the tools in terms of correct vs. incorrect split points, over- and undersplitting. Later, we will include an evaluation with regard to POS categories of the components wherever possible.

6.1.2 Results

According to the proposed methodology the first assessment of tool quality is achieved by a simple comparison of the output in the terms of:

- correct splits (when the splits provided by the tool either correspond to the morphological or structural gold splits, for example: *Bienenwachslasur* will result in the following gold splits: *Bienen*•*wachs*•*lasur* and *Bienenwachs*•*Lasur*);
- incorrect non-splits (when the tool perceives a word as a non-compound, a special form of undersplitting);
- wrong split points.

In this paper we present the result of such an analysis only for N+N type compounds (*Gerölllawine*, *Bombengeschäft*, *Tagblatt*), as well as for V+N type compounds (*Isolierschlauch*, *Meldeeinheit*, *Schleifgerät*), and also for certain types of tri-morphemic compounds (*Sperrholzrest*, *Heizkörpernische*, *Heißklebepistole*). The results obtained for binary compounds, N+N type (N = 626), are listed in Table 3.

Though CompoST clearly outperforms the other tools, some nouns still remain unsplit. Nevertheless it also made fewer wrong splits than SECOS or the TU-tool. The latter is almost as good as CompoST in terms of undersplitting, though it produced almost twice as many wrong splits. While SECOS made less mistakes with split points than the TU-tool, it was not as good as in distinguishing compounds from non-splittable items. One of the reasons for this performance might be the specialised nature of the data, as most of the N+N type compounds came from the domain of DIY instructions, such as: *Steinbearbeitung*, *Bohrmaschine*, *Drehzahl*. The results obtained for binary compounds of the V+N type (N = 118) are presented in Table 4.

In this case CompoST produced more nonsplits than the other tools, though its general performance is still higher than 65%, and only one compound was wrongly split (*Wegwerfgesellschaft*:

²<https://spacy.io/>

Tool	correct	non-split	wrong split
CompoST	582 (93%)	9 (1,4%)	35 (5,6%)
TU-tool	500 (80%)	15 (2,3%)	111 (17,7%)
Secos	496 (79%)	50 (7,8%)	79 (13,2%)

Table 3: Quantitative results on N+N compounds.

Tool	correct	non-split	wrong split
CompoST	78 (66%)	39 (33%)	1 (1%)
TU-tool	92 (78%)	2 (1,7%)	24 (20,3%)
Secos	75 (63,7%)	19 (16%)	24 (20,3%)

Table 4: Quantitative results on V+N compounds.

wrongly split as ??Weg•Werf•Gesellschaft instead of *Wegwerf•Gesellschaft*). The undersplitting tendency observed in N+N type compounds can be detected here as well. However the TU-tool outperforms the others with almost 78% of correct splits. The TU-tool and SECOS share the ca. 20% of wrong splits (??Ein•Lege•Bretter (TU-tool) and ??Einlegebre•Tter (SECOS) instead of *Einlege•Bretter*, ??Unter•Legscheibe (TU-tool) and ??Unter•legscheibe (SECOS) instead of *Unterleg•Scheibe*, ??Ans•Aug•Leistung (TU-tool) and ??Ansau•Gleis•Tung (SECOS) instead of *Ansaug•Leistung*). Examples of selected ternary compounds of different types ($N = 173$) are given in the table 5.

There may not be enough candidate data to assess all patterns, as A+N+N and V+N+N are rather rare in our texts; more data may be needed in the future to allow us to come up with a more meaningful evaluation. Nevertheless, both the TU-tool and SECOS provided consistently good results, with low percentages of wrong splits and almost no undersplitting. CompoST on the other hand exhibits a considerable amount of undersplitting, but produces only very few wrong splits. It remains unclear why A+N+N compounds lead to problems with CompoST. Our test set contained also non-compounds ($N = 86$), so that we could investigate oversplitting and the ability to distinguish compounds from other word formation products. The non-splittable candidates are mostly derivatives, some of which are phrasal derivatives:

- Derivation products: *Möglichkeit*, *Verschraubung*;
- Phrasal derivatives: *Rechtwinkligkeit*

The results are presented in Table 6.

Again CompoST clearly outperforms other tools in this task. It provides many good solutions and only a small amount of errors. Both the TU-tool and SECOS tend to produce erroneous splits in almost two thirds of the cases; their recognition capacity of non-splittable terms is thus not particularly good yet. All the three systems presented above were tested and their output was analyzed. Due to the underlying processing method the TU-tool and SECOS more often produce oversplitting of compounds (SECOS: ??W•Ärmer•Ückgew•Innungs•Anlage instead of *Wärme•Rückgewinnungs•Anlage*, ??Wasser•Rückgew•Innungs•Anlage instead of *Wasser•Ruckgewinnungs•Anlage*, and ??Un•Kennt•Lich•Machung instead of *Unkenntlichmachung*; TU-tool: ??Ver•Blend•Mauer•Werk instead of *Verblend•mauerwerk*, and ??Sch•Werst•Behinderten•Betreuung instead of *Schwerst•Behinderten•Betreuung*), while CompoST undersplits compounds from the general language even when the parameters are set to enforce splitting.

6.2 Post-training on domain-specific text data

For the evaluation of post-training CompoST, we take all word types from the DIY corpus as candidates for compound splitting, which are tagged as nouns. We both run the original CompoST (ORIG) and the version of CompoST adapted to the DIY domain (MOD). The results are shown in table 7. Overall, the modified version of CompoST finds more compounds than original CompoST does (first two rows of table). However, the difference is not big (259 compounds). Furthermore, for the majority of the cases, both splitter

Type	Tool	correct	non-split	wrong split
N + N + N (114) <i>Span•Holz•Platte</i>	CompoST	97 (85%)	14 (12,3%)	3 (2,7%)
	TU-tool	105 (92%)	0 (0%)	9 (8%)
	Secos	92 (81%)	3 (2,7%)	19 (16,3%)
A + N + N (35) <i>Rund•holz•stab</i>	CompoST	11 (31,4%)	21 (60%)	3 (8,4%)
	TU-tool	31 (89%)	0 (0%)	4 (11%)
	Secos	30 (86%)	0 (0%)	5 (14%)
V + N + N (25) <i>Senk•kopf•schraube</i>	CompoST	22 (88%)	3 (12%)	0 (0%)
	TU-tool	22 (88%)	0 (0%)	3 (12%)
	Secos	21 (84%)	0 (0%)	4 (16%)
All types (294)	CompoST	195 (66%)	91 (31%)	8 (3%)
	TU-tool	261 (89%)	3 (1%)	30 (10%)
	Secos	234 (80%)	8 (3%)	52 (17%)

Table 5: Quantitative results for selected ternary candidates.

Tool	correct	wrong split
CompoST	82 (95%)	4 (5%)
TU-tool	33 (38%)	53 (62%)
Secos	43 (50%)	43 (50%)

Table 6: Quantitative results on non-splittable items.

versions split identically (row 3), i.e. roughly 95% of the compounds split by MOD are split in the same way by ORIG. Rows 4 to 9 show the cases where the splitters do not agree, which is further analyzed below.

feature	#
all ORIG splits	59,936
all MOD splits	60,195
same split	57,145
only MOD splits	640
only ORIG splits	411
MOD more splits	232
ORIG more splits	227
different split points	127
lower/upper difference	1,793

Table 7: Comparison of the splitting results for the original CompoST (ORIG) and CompoST post-trained on a DIY corpus (MOD).

Only MOD splits vs. only ORIG splits. MOD splits more compounds than ORIG. In return, it misses compounds which were originally split (“only ORIG splits”). This makes up roughly 2/3 of the size of the compounds only split by MOD. It

seems likely that the missed compounds originate from general language, and the newly split ones are domain-specific. However, when analyzing the compounds, this is not the case; clear DIY-compounds like *Akkuschrauber* (“screwdriver”), *Stichsäge* (“padsaw”) or *Heimwerker* (“DIYer”) are not split by MOD.

Secondly, we want to analyze the impact of hyphenated compound candidates. An example would be *Douglasien-Bodendielen* (“douglas fir-floor boards”), where the split point is obvious because of the hyphen. There are rare cases where such a split would be wrong, e.g. 3-in-1 or 200-er. We throw out all compounds where the split point is set at the hyphen and show the result in table 8 (columns “only X splits”). Obviously, most compounds that MOD missed were hyphenated compounds; for closed compounds, MOD shows a superior performance for both binary and ternary compounds.

	only X splits		X more splits	
	ORIG	MOD	ORIG	MOD
binary	43	600	-	-
ternary	0	50	137	22
nary	-	-	9	0

Table 8: Difference of splitting results for the original CompoST (ORIG) and post-trained CompoST (MOD) with disregarding all compounds with splits at hyphens.

MOD more splits vs. ORIG more splits. In these cases, both splitters split the same compound but the number of splits is different. While for

the overall results (table 7) this part seems to be rather equally sized for the splitters, focusing on the closed, not hyphenated compounds again (table 8, columns “X more splits”) the picture is quite different. MOD produces fewer splits, i.e. contracts components within a compound. For example, ORIG splits *Schraubendreherklingen* (“screwdriver blades”) as *Schraube•Dreher•Klingen* (“screw•driver•blades”), while MOD splits *Schraubendreher•Klingen* (“screwdriver•blades”). We conclude that MOD finds some compounds to occur frequently and thus does not split them anymore. This intuition also coincides with the results from the previous paragraph, that DIY compounds like *Akkuschrauber* (“screwdriver”) are not split anymore by MOD.

Different split points. In these cases, both splitters split the same compound and return the same number of splits, but the split points are differently set. When analyzing the compounds, we find that in most cases the results are different because the modifier is either lemmatized as noun or verb, e.g. *Putz/putzen* (“plastering/to clean”), or the lemma is different: *Dosen* → *Dose/Dosis*. Some errors result from the Fugen-s (*Prozessor•Steuerung* “processor controlling” vs. ??*Prozessor•Teuerung*, lit.: “processor increase in prices”), or a completely wrong split. MOD performs superiorly to ORIG because it always selects the more likely lemma in the domain (e.g. *Putz* instead of *putzen*). We randomly select 30 compounds of this category and compare the splitting results; MOD splits 18 times correctly, ORIG only 8 times (in the other cases, both splits were incorrect).

Lower/upper difference. In these cases, both splitters split the same compound, return the same number of splits and find the same split points. Only upper- and the lowercasing is different. When analyzing the respective compound splits, one can see that it is mostly again the modifier which is different. Sometimes this is a discrepancy between verb and nominalized verb (e.g. *Sägetisch* “sawing table” is either split as *sägen•Tisch* “to saw•table” or *Sägen•Tisch* “sawing•table”), or upper- or lowercasing is just wrong (e.g. *Nahkontrolle* is split as *naht•Kontrolle* “joint examination”). It is unclear where this effect comes from. When again extracting 30 compounds randomly, MOD lemmatizes 15 times correctly, and ORIG lemmatizes 14 times correctly. To conclude, no splitter shows superior

performance here.

7 Discussion

In general, it is rather difficult to compare and evaluate the performance of different compound splitters. They return diverse splittings, e.g. they either return binary or n-ary splits, lemmatize the results or additionally POS-tag them. For some splitters, there even are several settings available (as for example, restricting either to a binary split or allowing an n-ary split). Thus, sometimes a comparison can be hard. For example, do we prefer a splitter that does not lemmatize against a splitter that lemmatizes, but sometimes returns wrong lemmas? Finally, the follow-up task for the compound splitting might decide which splitter we will use.

8 Conclusion and outlook

We presented a two-part study to evaluate the performance of German compound splitters on noun compounds, for general language and for specific domains. In a first experiment, we conducted a gold-standard-based evaluation of three compound splitters on general-language and domain-specific compounds. The splitters are CompoST, SECOS and a CRF-based tool from University of Tübingen. We explained data sampling from specialized corpora and from an inventory of general language phenomena in compounding. We noted that CompoST tends to undersplit compounds (likely due to a lack of lexical knowledge in SMOR), while the other two tools tend to oversplit. Consequently, CompoST also performs best on non-splittable items (95% correct vs. 50% for the second best tool). Its precision is highest for N+N compounds. TU-Tool produces more correct splits on V+N compounds, but also produces more incorrect splits. It is the best-performing tool on tri-morphemic noun compounds, with SECOS being second and CompoST last (only 66% correct vs. 89% with TU-Tool). TU-Tool produces a slightly higher amount of wrong splits than CompoST for tri-morphemic compounds, but therefore CompoST does not split nearly one third of the compounds. In general, CompoST rarely produces splits the result of which are non-morphemic letter sequences (in contrast to *Einlegebre-Tter* discussed in section 6.1.2).

In a second experiment, we post-trained CompoST on domain-specific DIY data, and compared the results for splitting domain-specific compounds. We found that for roughly 95% of the compound

candidates, the original and the modified splitter return identical splits. For the rest of the compounds, we performed a detailed evaluation with respect to several features, like the number of splits or a difference of the exact split points. We find that in these cases the adapted CompoST mostly outperforms the original one, especially for binary and ternary closed compounds. This qualitative improvement is quantitatively watered down by the fact that the original CompoST more often splits hyphenated compound candidates than the post-trained version. The modified version more often contracts components within an n-ary compound, presumably due to the increased number of occurrences of a complex component (e.g. *Heimwerker*) in the data used for post-training.

Overall, the comparison of compound splitters proved to be more difficult than one would expect, as the tools come with widely diverging features: some tools only provide one split-point, others do not come with training data, yet others include lemmatization of the output, which in some cases can be a source of further errors. Against this background, we see a need for further detailed methodological work on the topic.

References

- Fabienne Cap. 2014. *Morphological processing of compounds for statistical machine translation*. Ph.D. thesis.
- Joachim Daiber, Lautaro Quiroz, Roger Wechsler, and Stella Frank. 2015. Splitting compounds by semantic analogy. In *Proceedings of the 1st Deep Machine Translation Workshop*, pages 20–28, Praha, Czechia. ÚFAL MFF UK.
- Elke Donalies. 2011. Basiswissen deutsche wortbildung. 2., überarbeitete auflage. *Tübingen/Basel: Francke*.
- Elke Donalies. 2014. Morphologie: Morpheme, wörter, wortbildungen. *Ossner, Jakob/Zinsmeister, Heike (Hrsg.): Sprachwissenschaft fr das Lehramt*, pages 157–180.
- Carla Parra Escartín. 2014. Chasing the perfect splitter: A comparison of different compound splitting tools. In *LREC*, pages 3340–3347.
- Gertrud Faaß and Kerstin Eckart. 2013. Sdewac—a corpus of parseable sentences from the web. In *Language processing and knowledge in the Web*, pages 61–68. Springer.
- Wolfgang Fleischer and Irmhild Barz. 1995. *Wortbildung der deutschen Gegenwartssprache*.
- Roland Hauser. 1994. Results of the 1. morphlympics. *LDV-FORUM*.
- Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.
- Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *10th Conference of the European Chapter of the Association for Computational Linguistics*.
- Jianqiang Ma, Verena Henrich, and Erhard Hinrichs. 2016. Letter sequence labeling for compound splitting. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 76–81.
- Lieve Macken and Arda Tezcan. 2018. Dutch compound splitting for bilingual terminology extraction. *Multiword Units in Machine Translation and Translation Technology*, 341.
- Lorelies Ortner, Elgin Müller-Bollhagen, Hanspeter Ortner, Hans Wellmann, Maria Pümpel-Mader, and Hildegard Gärtnert. 1991. *Deutsche Wortbildung. Typen und Tendenzen in der Gegenwartssprache. Vierter Hauptteil: Substantivkomposita*. Berlin, New York: De Gruyter.
- Maria Pümpel-Mader, Elsbeth Gassner-Koch, Hans Wellmann, and Lorelies Ortner. 1992. *Deutsche Wortbildung: Typen und Tendenzen in der Gegenwartssprache; eine Bestandsaufnahme des Instituts für Deutsche Sprache, Forschungsstelle Innsbruck. Hauptteil 5. Adjektivkomposita und Partizipialbildungen*. Berlin, New York: de Gruyter.
- Martin Riedl and Chris Biemann. 2016. Unsupervised compound splitting with distributional semantics rivals supervised methods. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 617–622.
- Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. Smor: A German computational morphology covering derivation, composition and inflection. In *LREC*, pages 1–263. Lisbon.
- Don Tuggener. 2016. *Incremental coreference resolution for German*. Ph.D. thesis, Universität Zürich.
- Marion Weller-Di Marco. 2017. Simple compound splitting for German. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 161–166.
- Patrick Ziering, Stefan Müller, and Lonneke van der Plas. 2016. Top a splitter: Using distributional semantics for improving compound splitting. In *Proceedings of the 12th Workshop on Multiword Expressions*, pages 50–55.

Neural classification with attention assessment of the implicit-association test OMT and prediction of subsequent academic success

Dirk Johannßen

MIN Faculty,

Dept. of Computer Science

Universität Hamburg

& Nordakademie

<http://lt.informatik.uni-hamburg.de/>

{biemann, johannssen}@informatik.uni-hamburg.de

Abstract

Operant motives are unconscious intrinsic desires that can be measured by implicit methods, such as the Operant Motive Test (OMT) employs. During the OMT, participants are asked to write freely associated texts to provided questions and images. Trained psychologists label these textual answers with one of four motives. The identified motives allow for psychologists to predict behavior, long-term development, and subsequent success. We use a long short-term memory neural network (LSTM) combined with an attention mechanism for classification of OMT textual answers and show state-of-the-art performance over previous work. When investigating tokens that have high associated attention weights with the Linguistic Inquiry and Word Count (LIWC) tool, we find a weak connection between LIWC categories and the OMT theory. Lastly, we automatically annotate and count motives per participant and correlate counts with academic grades, finding a weak correlation between certain motives and subsequent academic success.

1 Introduction

The goal of our research is to classify psychometric textual data. Furthermore, we aim to investigate algorithmic decision making and validate automatic annotation by predictions in accordance with the psychometric theory. To pursue this goal, we perform multi-label classification on the Operant Motive Test (OMT, Section 2) with four labels. During this OMT, participants textually answer questions on images such as displayed in Figure 1 to provided questions.

Recent advances in artificial neural network architectures have established mechanisms that allow

Chris Biemann

MIN Faculty,

Dept. of Computer Science

Universität Hamburg

22527 Hamburg, Germany

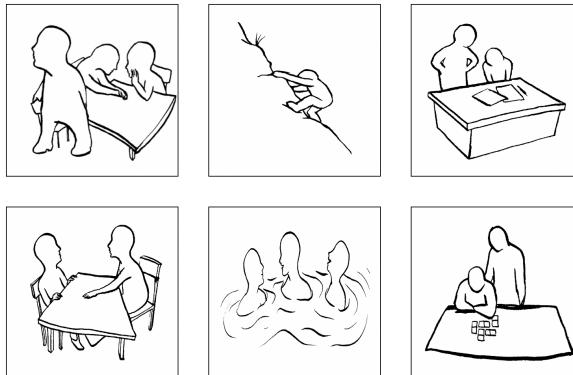


Figure 1: Some examples of images to be interpreted by participants utilized for the operant motive test (OMT). Exemplary answers given in Listing 1 correspond to the first picture. (Kuhl and Scheffer, 1999).

researchers to, in a limited fashion, inspect reasons for algorithmic decisions. One of these mechanisms is called *attention* and was found by Young et al. (2018) to be among the most broadly investigated and adopted elements of deep neural machine learning. We want to investigate access to algorithmic decision making by employing this attention mechanism (Section 3).

Lastly, the OMT theory states that some labeled motives allow for predictions of subsequent academic success, which we inspect by counting annotated labels and correlating these counts with participant’s academic grades.

Even though there is a high demand for the automation of psychological textual data analysis (NLPsych), comparably little research has been performed on this interdisciplinary task (Johannßen and Biemann, 2018). Reasons for this circumstance include the lack of available labeled psychological text data, as Husseini Orabi et al. (2018) point out, and the mere difficulty of capturing psychological traits solemnly from texts, especially short texts. Since first, psychologists are skilled

workers for such a labeling task and secondly, said task is difficult, labeling such psychometric textual data is costly. Also, interpretability and transparency are crucial for gaining insights into the nature of some tasks including security, medicine, and psychology, which is often more valuable for researchers than reaching the highest classification performance scores (Zhang et al., 2018).

In this work, we focus on the following research questions: i) Do neural architectures outperform a previous non-neural machine learning approach and if so, which architectures perform how well? ii) Do the attention weights matter and reveal any insights into algorithmic decision making? iii) Is there a correlation between automatically predicted motives and subsequent academic success?

We describe the OMT in Section 2. Thereafter, we will discuss related work in Section 3. Section 4 describes the data basis of this work and its characteristics. Our research methodology will be described in Section 5. Results will be presented in Section 6. Finally, a conclusion will be drawn in Section 7.

2 Operant Motive Test

Implicit or operant motives are unconscious intrinsic desires, which can be measured by psychological implicit methods, which require participants to use introspection for the assessment of psychological attributes (Gawronski and De Houwer, 2014). During the testing procedure, participants are asked to write freely associated texts to provided questions and images. The OMT is such a test and emerged from the Thematic Apperception Test (TAT, (Murray, 1943)).

Listing 1 displays a few of the training instances that correspond to the first picture of Figure 1, which displays some examples out of several. Those images show one or multiple persons often in unclear scenarios and situations. Applicants are asked to answer four questions: i) What is important for the persons in this situation and what is s/he doing? ii) What is the person feeling? iii) Why does the person feel this way? iv) How does the story end? The four answers are concatenated to a single string. On this string, it is possible to annotate one of the three motives a) Affiliation (German 'Anbindung', letter A), b) Achievement (German 'Leistung', letter L) and c) Power (German 'Macht', letter M). The very first observed motive applies to the whole string, which is the

so-called primacy rule (Kuhl and Scheffer, 1999). Once participants express a motive, this motive is saturated. Therefore, the following motives ought to be ignored when analyzing the answers. If no motive can be identified, a zero will be annotated (the so-called zero rule).

A sie nimmt am Gespräch nicht teil und wendet sich ab. gelangweilt. es interessiert sie nicht, worüber die anderen beiden reden. schlecht.	M weicht ängstlich zurück. unterlegen. wird zurechtgewiesen.
Gelegenheit den Fehler zu korrigieren	
----- Translation -----	
A she does not take part in the conversation and turns away. bored.	
She does not care what the other two are talking about. Bad.	
M withdraws anxiously. Inferior.	
is rebuked. Opportunity to correct the mistake.	

Listing 1: German text examples of OMT answers with A being Affiliation and M being the power motive. The texts correspond to the first picture of Figure 1. Translations into English provided by the authors.

Implicit motives allow for the prediction of clinically measured non-verbal interpersonal communication such as the amount of smiling, laughing or eye contact (McAdams et al., 1984) as well as the job performance (Lang et al., 2012). Scheffer (2004) was able to show a significant ($p < 0.02$) multiple regression correlation with a negative beta slope (hence the lower the German grade, the better with 1 being *very good* and 5 *having failed*) between the achievement motive and z-standardized average grades of students from different departments.

3 Related Work

Previous approaches to predicting psychological traits. So far, approaches to psychological traits identification from texts often examined the connection between language and mental diseases. Current research mostly focuses on e.g. the detection of dementia (Masrani et al., 2017), crises (Demasi et al., 2019), suicide risks (Matero et al., 2019), mental illnesses (Zomick et al., 2019) or anxiety (Shen and Rudzicz, 2017) by the use of some form of natural language processing.

Nonetheless, some findings focus on motivation, success or characteristics. Tomasello (2002) describes the psychology of language as the method of focusing on the way people express themselves

rather than to focus on what meaning is conveyed.

Linguistic Inquiry and Word Count (LIWC) is a tool developed by Pennebaker et al. (1999) for text analysis, that utilizes previously validated categories containing word lists for which the membership ratio of an input sequence is being asserted. Furthermore, the tool calculates statistical values e.g. the average word length, the average count of word per sentence or the frequency of words longer than 6 characters. LIWC can be considered to be a standard tool for the analysis of texts from the psychological domain due to its broad utilization among researchers (Johannßen and Biemann, 2018). The German version of LIWC has been developed by Wolf et al. (2008).

So-called closed-class words are by far more informative than open-class words in terms of psychological language research. Closed-class words are words that tend to not change over centuries, which can be e.g. pronouns, prepositions or adverbs. Open-class words, on the other hand, are words that are strongly influenced by the time being, such as historical events or names. Pennebaker et al. (2014) found a link between the usage of closed-class words and academic success. During the study, which used the LIWC tool on written essays of college applicants and connected these to subsequent academic success, the authors showed that the rate of closed-class words are significantly ($p < 0.01$) positively correlated to subsequent academic success, regardless of the chosen essay topic or sought major.

In (Johannßen et al., 2019) we engineered hand-crafted features to train a logistic model tree (LMT, Landwehr et al. (2005)) for classifying the operant motives. An LMT is a decision tree, which performs logistic regressions at its leaves. The LMT model reached an F-score of 80.1. The perplexities of language models for each motive, closed-class words, and ratios (words per sentence ratio, type-token ratio) were the main features for classification decisions.

Deep learning. Since assessing psychological traits solemnly from language is a challenging task, many researchers circumvent this bottleneck by including further personal information e.g. from social media platforms (Souri et al., 2018). Husseini Orabi et al. (2018) adapted this approach when they employed convolutional neural networks (CNN, Le-Cun et al. (1998)) and recurrent neural networks (RNN) in combination with further information

from social media as labels such as average age, gender or posting frequency to enhance the detection of mental disorders.

In order to detect crises, Kshirsagar et al. (2017) combined neural and non-neural techniques. The data was obtained from the anonymous emotional support network Koko¹, which is available through multiple messaging applications.

A long short-term memory neural network (LSTM, (Hochreiter and Schmidhuber, 1997)) is a type of RNN which, in turn, is a deep neural network architecture, that allows for the neural cells to access other cells of the same recurrent layer with a time delay and thus develop a so-called memory. An LSTM furthermore employs memory cells that allow storing information of an arbitrary time horizon. Forget and update gates allow for these cells to purposely omit information and control, how the memory is altered. LSTMs have successfully solved the issues of vanishing or exploding gradients present in general RNNs (Hochreiter, 1998) and have been utilized for classifying short texts.

Lai et al. (2015) designed a recurrent convolutional neural network (RCNN) for text classification with promising results. An RCNN is an RNN with a max-pooling layer as its output. The main advantages of an RCNN in comparison with RNNs is the enhanced selection of targets or regions to have an impact on algorithmic decision making.

Young et al. (2018) found attention mechanisms as part of decoder-encoder-architectures to be amongst these recent advancements in their survey. Accordingly, attention mechanisms allow for decoders to assess their memory by referring back to their input sequence, which can enhance the network's performance. The idea of employing attention to a sequence-to-sequence (Seq2Seq) encoder-decoder system originated from Bahadanau et al. (2015).

With a sequence of annotations h_i being (h_1, \dots, h_{T_x}) , a context vector c_i represents the weighted sum of the annotations via:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \quad (1)$$

The weights α_{ij} are computed as:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \quad (2)$$

¹<https://itskoko.com/>

whilst $e_{ij} = a(s_{i-1}, h_j)$, with $a(\dots)$ being a score function describing how well two words are aligned.

In other words, the system encodes an input sequence (this could be e.g. a certain language or a whole text to be summarized) into a context vector. This context vector together with hidden states functions as input for the attention mechanism, which computes attention weights and passes this context vector together with the attention weights on to the output layer. This process is illustrated in Figure 2.

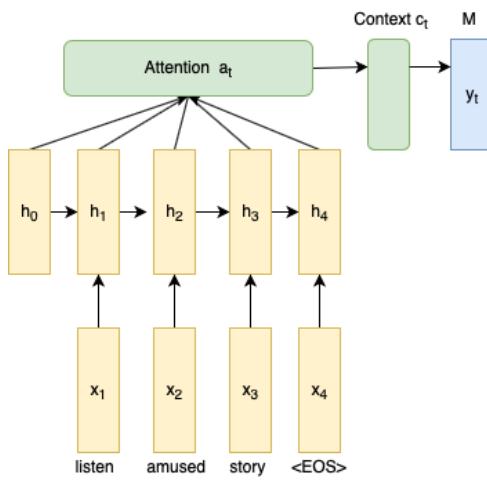


Figure 2: Illustration of the LSTM with attention mechanism. The LSTM receives hidden states and attention weights as inputs in order to output a corresponding context vector, which thereafter gets fed to a softmax output layer.

Attention mechanisms were successfully employed for various tasks. Gupta et al. (2018) utilized a CNN on group images for learning the global representation of the image and employed an attention mechanism for merging faces in order to learn local representations of only the faces, thus leading to a network capable of detecting emotions from entire groups of people. For this, the authors employed a Seq2Seq system with attention mechanism (the additional attention mechanism was proposed by Vaswani et al. (2017)). Images received automated descriptions by using a CNN encoder, an attention layer, and an LSTM decoder by Xu et al. (2015). Furthermore, the authors were able to project the attention weights onto the images, visualizing the gaze of the network. Speech has been analyzed for detecting emotions utilizing an attention mechanism by Ramet et al. (2018).

On textual data, attention mechanisms have enhanced the performance of classification and comprehension tasks. Hermann et al. (2015) advanced automated reading comprehension and question answering for texts with minimal prior knowledge. So-called self-attention was the enabler of semantic role labeling (SRL) for Tan et al. (2018). Self-attention is a special case of an attention mechanism, that only requires a single sequence to compute its representation. Vinyals et al. (2015) showed that a Seq2Seq model with attention mechanism could enhance syntactic constituency parsing to state-of-the-art performance.

A small subset of this data was annotated by utilizing attention over words. The authors were able to find the explanation of depressions from texts with a performance as well as human annotators had, which the authors refer to as *gold explanation*.

On the contrary, recent studies have questioned the interpretability of attention weights and suggested not to equate attention with explanation (Jain and Wallace, 2019). The authors found that if attention weights contribute to algorithmic decision making, the shuffling of these weights should significantly worsen results.

4 Data

The available data set has been collected and hand-labeled by researchers of the University of Trier. More than 14,600 volunteers participated in answering the OMT questions described in Section 2 to 15 provided images such as displayed in Figure 1. These participants produced 220,859 unique answers. Each answer was labeled by psychologists, which were trained with the OMT manual by Kuhl and Scheffer (1999). After pre-processing and cleaning the data, 209,716 text instances remain. The test and development set both constitute 10% of the available data, which is 20,960 instances each. The amount of motives in the available data is unbalanced with power (M) being by far the most frequent with 59%, achievement (L) constituting 19% of the data, affiliation (A) 17% and zero 5% (shown in Table 2 and in Figure 3). The pairwise annotator intraclass correlation was $r = .85$ on the Winter scale (Winter, 1994).

5 Methodology

Our methodology can be divided into two parts: the first is a natural language processing (NLP) task, which addresses research questions i) and ii)

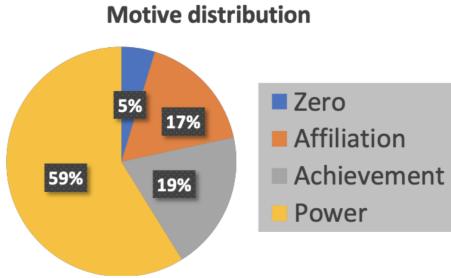


Figure 3: Graphical representation of the unevenly distributed motive labels amongst the data set.

and the second task answers research question iii) by counting classified motives per participant and correlating this count to academic grades.

In order to test whether an LSTM with an attention mechanism succeeds in outperforming the former best model for classifying the OMT, we employ the approach by Xu et al. (2015) on an already existing code basis for multiple text classifiers, which is utilized for further benchmarks as well.²

As for the word representations, we employed pre-trained fastText word embeddings for German (Bojanowski et al., 2017), provided by the developers.³ In contrast to Word2Vec word embeddings by Mikolov et al. (2013), fastText has the capability of representing tokens not included in the embedded words on the basis of character n-grams. The OMT data (described in Section 4) is noisy, has many spelling mistakes and would probably not sufficiently be represented by word-based embeddings.

5.1 Benchmarking systems

To our knowledge, psychometrics closely related to the TAT have not been classified with neural methods yet. The only classification on the OMT has been performed by utilizing an LMT model in our previous work (2019), which we compare to our neural approach. In order to put different architectures into perspective and to explore the relationship of our proposed LSTM system with attention mechanism, we performed multiple benchmarking experiments on the task of automatically assigning the four classes of operant motives described in Section 2 and thus aim to answer the second research question of how well other neural

approaches perform in comparison.

For this, we employed the following neural architectures, as reviewed in Section 3: LSTM, CNN, RNN, RCNN, Bi-LSTM with self-attention, LSTM with attention and Seq2One (a Seq2Seq variant with only one label as output) with attention. Since neural approaches are non-deterministic (Lai et al., 2015), we trained each model three times and averaged the F-scores for a stable assessment of results.

Three modifications of the LSTM with attention mechanism are employed: Firstly, we shuffled the attention weights before they got applied to the hidden states. Secondly, we reversed the direction of the input sequence to honor the OMT primacy rule. If this rule is followed and processing order has an influence, processing from right-to-left and classifying on the entire representation could improve results since the most influential signal (the first motive in the text) is accumulated last into the representation. Thirdly, we add comparable hand-crafted features as a fully connected input to the final classification softmax layer (e.g. part-of-speech (POS) tags, LIWC categories or the perplexities of trained language models per target motive), following Johannßen et al. (2019) to investigate in how far neural feature induction subsumes these features.

5.2 Psychometric predictions

After benchmarking, we utilize the most promising system for predictions in accordance with the OMT theory. 103 participating students answered the questions to 15 images, resulting in 1,545 answer sequences. Further, the data collection includes the grade of their bachelor’s thesis, which was completed a few years after the OMT was taken. We employ the experimental design of our previous work (Johannßen et al., 2019) to ensure a fair comparison. For this, we predict the motives of each of the 15 answers given per participant, count the appearances per motive and correlate these to the bachelor’s thesis grade.

5.3 Model training

All parameters of the models were tuned on a development set. Different fixed input sizes were considered for every architecture: Firstly we considered a fixed input length of 81 since the longest answer contains 81 words. Secondly, the average answer contains 20 words, which we considered as fixed input size in order to take the primacy rule (Section 2) into account. Shorter answers than

²<https://github.com/prakashpandey9/Text-Classification-Pytorch/tree/master/>

³Facebook’s AI Research, <https://fasttext.cc>

the fixed input length receive the padding token (<pad>), longer ones were truncated. Human annotators are asked to ignore the rest of a sequence after a very first motive could be identified. Terms not observed in the training vocabulary were replaced by an out-of-vocab (OOV) token. Dropouts of 0.3, 0.5 and 0.8 were evaluated, whereas 0.5 has shown to perform best for the RNNs and has also been suggested by Hinton et al. (2012). The number of iterations was set to 3,600 in 32 batches and two epochs. The models received word embedded fastText inputs with 100 and 300 dimensions, of which the 300-dimensional embeddings reached better results, and had two hidden layers with 256 cells each. Learning rates were set to 0.0001, 0.001 and 0.01 for each model, with 0.001 performing best. All results are displayed in Table 1 and were achieved with these unified best-performing parameters.

As for the LSTM with attention mechanism, which has shown to perform best, the model converged quickly to a loss of approx. 0.4 and oscillates thereafter.

5.4 Attention weights assessment

As shown by Vaswani et al. (2017), the attention mechanism (described in Section 3) has broadly been believed to contribute to explainable artificial intelligence by shedding light on algorithmic decision making. Many authors have followed the initial idea and e.g. applied heat maps according to attention weights for input sequences and investigated algorithmic decision making. Other studies find contrary evidence that attention weights do not necessarily reflect true meaning (Jain and Wallace, 2019). Even though we are aware of these controversies and limitations, we follow the critic’s suggestion to investigate whether attention weights make a difference in the performance of a system. For this, we measure on which index the most attention weight mass is accumulated. We hypothesized that this might often be the last token since attention weights usually traverse a sequence *in search* (metaphorically speaking) for suiting candidates and mostly does not find any of such, applying the most of the available attention weight to the last possible candidate – the last token. We will further collect sequences that do not show this behavior and thus have the largest attention weight mass assigned to other tokens than the last one. These tokens will be evaluated with the LIWC tool.

We would expect the motives to be reflected in the LIWC categories if they meant anything at all. We automatically assembled all classified instances, whose highest attention weight did not assemble on the very last token, exceeded 0.3 and was classified correctly.

6 Results

6.1 Model performance

Table 1 shows classification performance of the different approaches on the test set. We were able to improve over our previous classifier (Johannßen et al., 2019). Even though neural approaches often perform better than earlier machine learning (Zhang et al., 2018), only the results of the best-performing model, the LSTM with an attention mechanism, outperforms the feature-engineered LMT classification model by an F-score of 81.55 (the LMT scored 81.10 and thus only slightly worse) with a fixed input size of 20 tokens. The same model with the fixed size of the longest answer of 81 tokens performed worse with an F-score of 80.71 (not shown in Table 1). The other approaches, also with a fixed input size of 20 tokens, performed worse, mostly around a 79 F-score except for the CNN. Including 129 hand-crafted features, reversing the reading direction and shuffling attention weights did not improve the results, thus indicating that firstly, attention matters, secondly, the direction of classification is not as important and thirdly, the LSTM attention model learns the features (POS, LIWC categories, perplexity) incidentally. The confusion matrix of the best-performing model is displayed in Table 2. The same LSTM with attention mechanism enriched by similar hand-crafted features does not improve results further, indicating that the information from these features is subsumed by the induced representations. The inversion of the input sequence resulted in lower scores, indicating that either the model cannot make use of seeing earlier tokens later to account for the primacy rule, or that the primacy rule has not been followed consequently during annotation. Shuffling of the attention weights worsens the results, indicating that these weights matter for the classification task.

6.2 Assessment of the attention weights

Table 1 shows that the LSTM with attention mechanism scored significantly lower when its attention weights were shuffled compared to the one with

Model	\varnothing Accuracy	\varnothing Precision	\varnothing Recall	\varnothing F-score	$F \sigma$
CNN	63.26	59.34	63.62	61.41	2.36
RNN	68.73	73.10	68.73	70.85	1.59
LSTM	77.84	78.05	77.84	77.92	0.65
Sequence to One (Seq2One) with attention	77.34	76.81	77.43	77.12	1.53
LSTM Attn with shuffled attention weights	79.03	78.05	79.03	78.54	0.13
RCNN	79.70	79.35	79.81	79.58	0.77
Bi-LSTM with self-attention	81.16	80.35	81.16	80.75	0.31
LSTM Attn with 129 addit. handcrafted features	80.85	79.86	80.86	80.35	1.23
LSTM Attn with a reversed direction	80.87	80.05	80.87	80.46	0.99
LSTM with an attention mechanism (LSTM Attn)	81.94	81.15	81.96	81.55	0.09
LMT with 129 handcrafted features (baseline)	81.56	80.90	81.60	81.10	0.00

Table 1: Performance comparison between the LMT and neural systems. All models classified with a fixed input size of 20 tokens. The only system overcoming the strong baseline of the feature-based LMT is an LSTM with attention mechanism. This system was also tested in reversed direction, with shuffled attention weights and with 129 additional handcrafted features, all of which performed worse than the best model. We averaged all scores (\varnothing) from three trained models each, and provide the standard deviation across runs (σ).

		Predicted				
		0 5%	A 17%	L 19%	M 59%	Σ 100%
Actual	0	283	102	150	478	1,013
	A	29	2,739	112	646	3,526
	L	90	91	3,079	872	4,132
	M	126	657	404	11,102	12,289
	Σ	528	3,589	3,745	13,098	20,960

Table 2: The relative motive amounts and confusion matrix of the best performing system (LSTM Attn).

properly trained attention and assigned weights. Jain and Wallace (2019) stated that this case had occurred only rarely in their experiments, but that if this circumstance holds true, they would assume that attention weights could be considered for interpretation and explanation.

We can observe that on average, 79.85% of the available attention weight mass was assigned to the very last token of each instance. It appears that the mechanism considered one token at a time from left to right and determines whether attention weight mass should be assigned to the token in question. If this is not the case, the attention weight mass is being kept and the successor token is considered. When the mechanism reaches the end of the sequence, it assigns whatever attention weight mass is left to the very last token. The second and third index with the highest following attention weight masses are the second last and third last tokens re-

spectively. According to the OMT theory, the last tokens of a sequence, in general, should not provide the main information for encoding the whole sequence due to the primacy rule, this high attention weight mass on the last token indicates, that for the majority of classified instances, the attention weights do not serve as a widely applicable means to interpret the reasons for classification decisions in this setup.

Besides these last tokens, we aimed to investigate the mechanism further and compare these non-concluding tokens to all tokens by automatically assembling instances and attention weights.

Table 3 compares the four most prominent psychologically validated LIWC category memberships in percent per motive of all tokens versus non-final tokens with high attention weight masses. Most of the LIWC category names appear to be representative for the wordlists that they consist of. E.g. *positive emotion* consists of e.g. *love, nice and sweet*.

According to the OMT theory, people with a strong achievement motive desire intrinsic excellence. They tend to analyze problems thoroughly and focus on tasks. This description is reflected by *cognitive mechanism* that is almost twice as present for high attention mass tokens as it is for all tokens (27.39% compared to 14.11%). The categories *occupation* (e.g. observe, conduct, advancing) with 24.66% and *achieve* – already with the same name as the OMT motive – with 23.28% are high in presence as well. Compared to rather low *social*,

High attention weight mass			All tokens		
LIWC	per cent	words	LIWC	per cent	words
cognitive mechanism	27.39	intense concentrated motivated capabilities	social	15.17	-
occupation	24.66		cognitive mechanism	14.11	-
achieve	23.28		other references	11.44	-
insight	10.96		affect	10.49	-
Affiliation					
affect	12.12	important secure partner interested	social	19.76	-
positive emotion	12.12		other references	12.04	-
humans	9.09		affect	10.31	-
social	9.09		cognitive mechanism	9.48	-
Power					
affect	33.95	can feels dominant humiliated	social	18.99	-
cognitive mechanism	28.91		cognitive mechanism	11.46	-
positive emotion	24.93		other references	11.25	-
insight	20.16		affect	9.91	-

Table 3: LIWC analysis of tokens that received the most attention weight mass on the left with all tokens on the right separated by predicted labels (left) versus manually annotated labels (right).

affect and other references, the OMT theory for the achievement motive appears to be better represented by tokens with high attention. Single words include *intense, concentrated, motivated and capabilities*.

Similarly, the LIWC categories for the affiliation motive are *affect, positive emotion, humans and social* for the left columns and apparently reflect the description of a desire to solve problems cooperatively, whilst avoiding conflicts. However, scores for LIWC categories are rather low at 12.12% and 9.09%. The social LIWC category is strongly present on the right column for all tokens with 19.76%, as well as *affect* with 12.04%. The other two LIWC categories of the right columns *other references* and *cognitive mechanism* do not appear to align well with the affiliation motive.

Even though the desire to influence and alter one's surrounding and fellow beings, the power motive can be identified by positive expressions as well as rather harsh ones. All LIWC categories of these columns on the left appear to align with the power motive, which are *affect* (33.95%), *cognitive mechanism* (28.91%), *positive emotion* (24.93%) and *insight* (20.16%). The corresponding LIWC categories for all tokens on the right columns correspond with the exception of *other references* but are comparably weaker.

This comparison shows that tokens with high attention mass per motive correspond to the OMT

theory e.g. occupation and insight for achievement, whilst all tokens do show some correspondence (e.g. social and affiliation), but in general, do not align well with the OMT theory. Interestingly, when removing the tokens (besides the last ones) that received the most attention weight mass and re-evaluating the answers with the LIWC tool to test the counterhypothesis that high-attention tokens do not reflect the classes, the categories shift to ones that do not correspond to the OMT theory.

gelangweilt <i>bored</i>	weil <i>because</i>	sie <i>she</i>	jeden <i>every</i>	tag <i>day</i>	0
geborgen <i>protected</i>	weil <i>because</i>	die <i>the</i>	andere <i>other</i>	person <i>person</i>	A
gefährdet <i>challenged</i>	will <i>wants</i>	das <i>the</i>	ziel <i>goal</i>	erreichen <i>to reach</i>	L
zu <i>to</i>	maßregeln <i>disciplin</i>	dominant <i>dominant</i>	die <i>the</i>	andere <i>other</i>	M

Table 4: Heatmap according to the attention weights displayed on four example snippets of OMT answers in German with their glossed translations and targets (A for affiliation, M for power and L for achievement).

Examples are given in Table 4, which displays some tokens highlighted, according to the token's attention weight masses. These examples do not reflect the whole data basis but illustrate a possible aid for understanding the task at hand and might help develop tool support for this task or related psychometrics.

6.3 Correlation with bachelor's thesis grades

As described in Section 5, in order to analyze the predictive power of motives, we count predicted motives and correlate these counts to academic grades. While we previously found a weak correlation of $r = -0.2$ between power motive counts and the bachelor's thesis grade, the experiment in this work revealed a correlation of $r = -0.25$ between the bachelor's thesis grade and the achievement motive in this work, i.e. the higher the achievement motive count, the better the German grade value (1 equals *good*, 5 equals *having failed*). The power motive is positively correlated with a small $r = 0.14$, i.e. the higher the power motive count, the worse the German grade. Figure 4 shows scatter plot displaying the counts of the power and achievement motives and the achieved bachelor's thesis grade.

This discrepancy of both model's predictions is anomalous. If both models performed comparably well on the same type of data, both mod-

els should reveal comparable correlations between counted motives and grades. The investigation of each model’s motive predictions per student shows that the LSTM with attention mechanism often assigns the power motive but never zero, whilst the LMT model assigns zero on 17.76% of all cases, indicating that the LMT model often did not predict any motive. Thus, even though the models behave comparably well on test data of the same origin as the training data, they differ in their algorithmic decision making on data from a different origin.

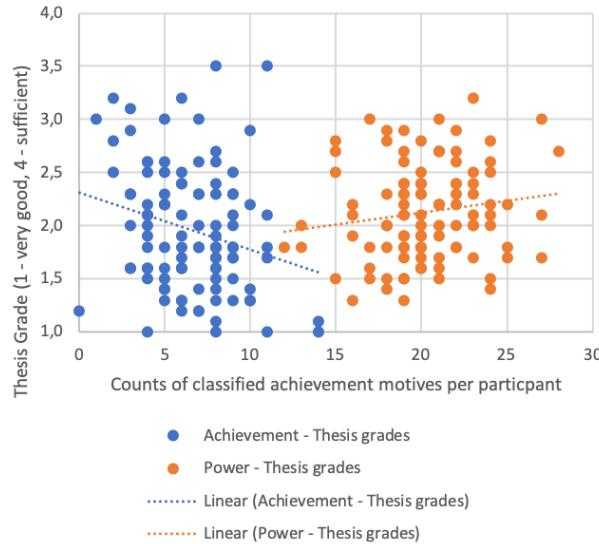


Figure 4: After predicting motives, the four motives per participants were counted. The power motive has the highest frequency. By counting predicted motives and correlating them to academic grades, a weak correlation of $r = -0.25$ could be observed between the achievement motive (blue dots) and the bachelor’s thesis grade (in Germany, the best grade is 1, reading: the higher the achievement motive count, the better the grade). In contrast, the plots shows that the higher the power motive counts (orange dots), the worse the grade with $r = 0.14$.

7 Conclusion and outlook

We were able to outperform prior classification of the OMT by employing an LSTM with an attention mechanism achieving an F-score of 81.55 and thus can positively answer research question i), asking whether our proposed model could outperform our former approach. Other architectures such as the RNN, LSTM, Bi-LSTM or the RCNN mostly reached an F-score of approx. 79. Attention weights only matter in thus far that the shuf-

fling of these weights worsens the results, asked by research question ii). The attention weight mass mostly accumulates on the very last token and thus does not allow for insights in the general case. For these cases where the attention weight mass was distributed among other tokens than the last one of a sequence, an analysis with the LIWC tool showed conformity of LIWC categories with the corresponding operant motives compared to these of all words. This indicates an overlap between the memberships per word of both linguistic assessments. This behavior of the highest attention mass on last tokens could be canceled out by employing a Bi-LSTM with attention mechanism and concatenating the attention weights of both systems, which we consider for future experiments. When removing these tokens and re-evaluating the sequence with the LIWC tool, the results shift, which has to be investigated further. Research question iii) questioned a correlation between identified motives and subsequent academic success as prior research has shown. This correlation could slightly be outperformed with $r = -0.25$ between the counted achievement motives and bachelor’s thesis grade, which is a weak correlation much different to former predictions of the LMT model that assigned zeros more often than the LSTM model with attention mechanism. Since zero marks indecisiveness, it can be assumed that the LMT model does not generalize as well as the LSTM – though this assumption would have to be further examined by e.g. having trained psychologists assess the outputs of both models. Furthermore, direct predictions from language to grades could be investigated, hence losing information at the intermediate step of automatically annotated motives.

Nonetheless, further validation is appropriate due to recent debates upon attention weights as indicators of interpretation. One approach for validation would be to provide trained psychologists for labeling the OMT with tokens that received comparably much attention weight mass and with tokens that did not measure how many cases would have been identified by said psychologists. Furthermore, we aim to provide annotators with a tool with attention-based highlighting for possibly saving time and expenses during the labeling process. Further numerical improvements could result from using contextualized embeddings, e.g. Bidirectional Encoder Representations from Transformers (BERT, Devlin et al. (2019)).

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR*, San Diego, CA, USA.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Orianna Demasi, Marti A. Hearst, and Benjamin Recht. 2019. Towards augmenting crisis counselor training by improving message retrieval. In *Proceedings of the Sixth Workshop on Computational Linguistics and Clinical Psychology*, pages 1–11, Minneapolis, MN, USA.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, MN, USA.
- Bertram Gawronski and Jan De Houwer. 2014. Implicit measures in social and personality psychology. *Handbook of research methods in social and personality psychology*, 2:283–310.
- Aarush Gupta, Dakshit Agrawal, Hardik Chauhan, Jose Dolz, and Marco Pedersoli. 2018. An attention model for group-level emotion recognition. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction, ICMI ’18*, pages 611–615, New York, NY, USA.
- Karl M. Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS’15, pages 1693–1701, Cambridge, MA, USA.
- Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation*, 9(8):1735–1780.
- Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.
- Ahmed Husseini Orabi, Prasadith Buddhitha, Mahmoud Husseini Orabi, and Diana Inkpen. 2018. Deep learning for depression detection of Twitter users. In *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, pages 88–97, New Orleans, LA, USA.
- Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, MN, USA.
- Dirk Johannßen and Chris Biemann. 2018. Between the Lines: Machine Learning for Prediction of Psychological Traits - A Survey. In *International Cross-Domain Conference, CD-MAKE*, pages 192–211, Hamburg, Germany.
- Dirk Johannßen, Chris Biemann, and David Scheffer. 2019. Reviving a psychometric measure: Classification of the Operant Motive Test. In *Proceedings of the Sixth Annual Workshop on Computational Linguistics and Clinical Psychology (CLPsych)*, pages 121–125, Minneapolis, MN, USA.
- Rohan Kshirsagar, Robert Morris, and Samuel Bowman. 2017. Detecting and explaining crisis. In *Proceedings of the Fourth Workshop on Computational Linguistics and Clinical Psychology — From Linguistic Signal to Clinical Reality*, pages 66–73, Vancouver, BC, Canada.
- Julius Kuhl and David Scheffer. 1999. *Der operante Multi-Motiv-Test (OMT): Manual [The operant multi-motive-test (OMT): Manual]*. Impart, Osnabrück, Germany: University of Osnabrück.
- Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*, pages 2267–2273, Austin, TX, USA.
- Niels Landwehr, Mark A. Hall, and Eibe Frank. 2005. Logistic Model Trees. *Machine Learning*, 59(1):161–205.
- Jonas W. B. Lang, Ingo Zettler, Christian Ewen, and Ute R. Hüscher. 2012. Implicit motives, explicit traits, and task and contextual performance at work. *The Journal of Applied Psychology*, 97(6):1201–1217.
- Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- Vaden Masrani, Gabriel Murray, Thalia Field, and Giuseppe Carenini. 2017. Detecting dementia through retrospective analysis of routine blog posts by bloggers with dementia. In *Proceedings of*

- the 16th Workshop on Biomedical Natural Language Processing*, pages 232–237, Vancouver, BC, Canada.
- Matthew Matero, Akash Idnani, Youngseo Son, Sal Giorgi, Huy Vu, Mohammad Zamani, Parth Limbachiya, Sharath C. Guntuku, and H. Andrew Schwartz. 2019. Suicide risk assessment with multi-level dual-context language and BERT. In *Proceedings of the Sixth Workshop on Computational Linguistics and Clinical Psychology*, pages 39–44, Minneapolis, MN, USA.
- Dan P. McAdams, R. Jeffrey Jackson, and Carol Kirshnit. 1984. Looking, laughing, and smiling in dyads as a function of intimacy motivation and reciprocity. *Journal of Personality*, 52(3):261–273.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Lake Tahoe, NV, USA.
- Henry A. Murray. 1943. *Thematic Apperception Test*. Harvard University Press.
- James W. Pennebaker, Martha E. Francis, and Roger J. Booth. 1999. Linguistic inquiry and word count (LIWC). *Software manual*. <http://liwc.wpengine.com>.
- James W. Pennebaker, Cindy K. Chung, Joey Frazee, Gary M. Lavergne, and David I. Beaver. 2014. When small words foretell academic success: The case of college admissions essays. *PLOS ONE*, 9(12):e115844.
- Gaetan Ramet, Philip N. Garner, Michael Baeriswyl, and Alexandros Lazaridis. 2018. Context-aware attention mechanism for speech emotion recognition. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 126–131, Athens, Greece.
- David Scheffer. 2004. *Implizite Motive: Entwicklung, Struktur und Messung [Implicit Motives: Development, Structure and Measurement]*. Hogrefe Verlag, Göttingen, Germany, 1st edition.
- Judy H. Shen and Frank Rudicicz. 2017. Detecting anxiety on Reddit. In *Proceedings of the Fourth Workshop on Computational Linguistics and Clinical Psychology — From Linguistic Signal to Clinical Reality*, pages 58–65, Vancouver, BC, Canada.
- Alireza Souri, Shafiqeh Hosseinpour, and Amir M. Rahmani. 2018. Personality classification based on profiles of social networks’ users and the five-factor model of personality. *Human-centric Computing and Information Sciences*, 8(1):24.
- Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4929–4936, New Orleans, LA, USA.
- Michael Tomasello. 2002. *The New Psychology of Language: Cognitive and Functional Approaches to Language Structure*. Psychology Press, 2nd edition.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Long Beach, CA, USA.
- Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS’15, pages 2773–2781, Cambridge, MA, USA.
- David Winter. 1994. *Manual for scoring motive imagery in running text*. Dept. of Psychology, University of Michigan (unpublished).
- Markus Wolf, Andrea B. Horn, Matthias R. Mehl, Severin Haug, James W. Pennebaker, and Hans Kordy. 2008. Computergestützte quantitative Textanalyse: Äquivalenz und Robustheit der deutschen Version des Linguistic Inquiry and Word Count. *Diagnostica*, 54(2):85–98.
- Kelvin Xu, Jimmy L. Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML’15, pages 2048–2057, Lille, France.
- Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent Trends in Deep Learning Based Natural Language Processing [Review Article]. *IEEE Computational Intelligence Magazine*, 13:55–75.
- Xinyang Zhang, Ningfei Wang, Shouling Ji, Hua Shen, and Ting Wang. 2018. Interpretable deep learning under fire. *CoRR*, abs/1812.00891.
- Jonathan Zomick, Sarah I. Levitan, and Mark Serper. 2019. Linguistic analysis of schizophrenia in Reddit posts. In *Proceedings of the Sixth Workshop on Computational Linguistics and Clinical Psychology*, pages 74–83, Minneapolis, MN, USA.

Towards Multimodal Emotion Recognition in German Speech Events in Cars using Transfer Learning

Deniz Cevher^{1,2*}, Sebastian Zepf^{1*} and Roman Klinger²

¹ Mercedes-Benz Research & Development, Daimler AG, Sindelfingen, Germany

² Institut für Maschinelle Sprachverarbeitung, University of Stuttgart, Germany

{firstname.lastname}@daimler.com

{firstname.lastname}@ims.uni-stuttgart.de

Abstract

The recognition of emotions by humans is a complex process which considers multiple interacting signals such as facial expressions and both prosody and semantic content of utterances. Commonly, research on automatic recognition of emotions is, with few exceptions, limited to one modality. We describe an in-car experiment for emotion recognition from speech interactions for three modalities: the audio signal of a spoken interaction, the visual signal of the driver’s face, and the manually transcribed content of utterances of the driver. We use off-the-shelf tools for emotion detection in audio and face and compare that to a neural transfer learning approach for emotion recognition from text which utilizes existing resources from other domains. We see that transfer learning enables models based on out-of-domain corpora to perform well. This method contributes up to 10 percentage points in F_1 , with up to 76 micro-average F_1 across the emotions joy, annoyance and insecurity. Our findings also indicate that off-the-shelf-tools analyzing face and audio are not ready yet for emotion detection in in-car speech interactions without further adjustments.

1 Introduction

Automatic emotion recognition is commonly understood as the task of assigning an emotion to a pre-defined instance, for example an utterance (as audio signal), an image (for instance with a depicted face), or a textual unit (e.g., a transcribed utterance, a sentence, or a Tweet). The set of emotions is often following the original definition by Ekman (1992), which includes anger, fear, disgust, sadness, joy,

and surprise, or the extension by Plutchik (1980) who adds trust and anticipation.

Most work in emotion detection is limited to one modality. Exceptions include Busso et al. (2004) and Sebe et al. (2005), who investigate multimodal approaches combining speech with facial information. Emotion recognition in speech can utilize semantic features as well (Anagnostopoulos et al., 2015). Note that the term “multimodal” is also used beyond the combination of vision, audio, and text. For example, Soleymani et al. (2012) use it to refer to the combination of electroencephalogram, pupillary response and gaze distance.

In this paper, we deal with the specific situation of car environments as a testbed for multimodal emotion recognition. This is an interesting environment since it is, to some degree, a controlled environment: Dialogue partners are limited in movement, the degrees of freedom for occurring events are limited, and several sensors which are useful for emotion recognition are already integrated in this setting. More specifically, we focus on emotion recognition from speech events in a dialogue with a human partner and with an intelligent agent.

Also from the application point of view, the domain is a relevant choice: Past research has shown that emotional intelligence is beneficial for human computer interaction. Properly processing emotions in interactions increases the engagement of users and can improve performance when a specific task is to be fulfilled (Klein et al., 2002; Coplan and Goldie, 2011; Partala and Surakka, 2004; Pantic et al., 2005). This is mostly based on the aspect that machines communicating with humans appear to be more trustworthy when they show empathy and are perceived as being natural (Partala and Surakka, 2004; Brave et al., 2005; Pantic et al., 2005).

Virtual agents play an increasingly important role in the automotive context and the speech modality is increasingly being used in cars due to its potential to limit distraction. It has been shown

*The first two authors contributed equally.

that adapting the in-car speech interaction system according to the drivers' emotional state can help to enhance security, performance as well as the overall driving experience (Nass et al., 2005; Harris and Nass, 2011).

With this paper, we investigate how each of the three considered modalities, namely facial expressions, utterances of a driver as an audio signal, and transcribed text contributes to the task of emotion recognition in in-car speech interactions. We focus on the five emotions of *joy*, *insecurity*, *annoyance*, *relaxation*, and *boredom* since terms corresponding to so-called fundamental emotions like *fear* have been shown to be associated to too strong emotional states than being appropriate for the in-car context (Dittrich and Zepf, 2019). Our first contribution is the description of the experimental setup for our data collection. Aiming to provoke specific emotions with situations which can occur in real-world driving scenarios and to induce speech interactions, the study was conducted in a driving simulator. Based on the collected data, we provide baseline predictions with off-the-shelf tools for face and speech emotion recognition and compare them to a neural network-based approach for emotion recognition from text. Our second contribution is the introduction of transfer learning to adapt models trained on established out-of-domain corpora to our use case. We work on German language, therefore the transfer consists of a domain and a language transfer.

2 Related Work

2.1 Facial Expressions

A common approach to encode emotions for facial expressions is the facial action coding system FACS (Ekman and Friesen, 1978; Sujono and Gunawan, 2015; Lien et al., 1998). As the reliability and reproducability of findings with this method have been critically discussed (Mesman et al., 2012), the trend has increasingly shifted to perform the recognition directly on images and videos, especially with deep learning. For instance, Jung et al. (2015) developed a model which considers temporal geometry features and temporal appearance features from image sequences. Kim et al. (2016) propose an ensemble of convolutional neural networks which outperforms isolated networks.

In the automotive domain, FACS is still popular. Ma et al. (2017) use support vector machines to distinguish *happy*, *bothered*, *confused*, and *con-*

centrated based on data from a natural driving environment. They found that *bothered* and *confused* are difficult to distinguish, while *happy* and *concentrated* are well identified. Aiming to reduce computational cost, Tews et al. (2011) apply a simple feature extraction using four dots in the face defining three facial areas. They analyze the variance of the three facial areas for the recognition of *happy*, *anger* and *neutral*. Ihme et al. (2018) aim at detecting *frustration* in a simulator environment. They induce the emotion with specific scenarios and a demanding secondary task and are able to associate specific face movements according to FACS. Paschero et al. (2012) use OpenCV (<https://opencv.org/>) to detect the eyes and the mouth region and track facial movements. They simulate different lightning conditions and apply a multilayer perceptron for the classification task of Ekman's set of fundamental emotions.

Overall, we found that studies using facial features usually focus on continuous driver monitoring, often in driver-only scenarios. In contrast, our work investigates the potential of emotion recognition during speech interactions.

2.2 Acoustic

Past research on emotion recognition from acoustics mainly concentrates on either feature selection or the development of appropriate classifiers. Rao et al. (2013) as well as Ververidis et al. (2004) compare local and global features in support vector machines. Next to such discriminative approaches, hidden Markov models are well-studied, however, there is no agreement on which feature-based classifier is most suitable (El Ayadi et al., 2011). Similar to the facial expression modality, recent efforts on applying deep learning have been increased for acoustic speech processing. For instance, Lee and Tashev (2015) use a recurrent neural network and Palaz et al. (2015) apply a convolutional neural network to the raw speech signal. Neumann and Vu (2017) as well as Trigeorgis et al. (2016) analyze the importance of features in the context of deep learning-based emotion recognition.

In the automotive sector, Boril et al. (2011) approach the detection of negative emotional states within interactions between driver and co-driver as well as in calls of the driver towards the automated spoken dialogue system. Using real-world driving data, they find that the combination of acoustic features and their respective Gaussian mixture model

scores performs best. Schuller et al. (2006) collects 2,000 dialog turns directed towards an automotive user interface and investigate the classification of *anger*, *confusion*, and *neutral*. They show that automatic feature generation and feature selection boost the performance of an SVM-based classifier. Further, they analyze the performance under systematically added noise and develop methods to mitigate negative effects. For more details, we refer the reader to the survey by Schuller (2018). In this work, we explore the straight-forward application of domain independent software to an in-car scenario without domain-specific adaptations.

2.3 Text

Previous work on emotion analysis in natural language processing focuses either on resource creation or on emotion classification for a specific task and domain. On the side of resource creation, the early and influential work of Pennebaker et al. (2015) is a dictionary of words being associated with different psychologically relevant categories, including a subset of emotions. Another popular resource is the NRC dictionary by Mohammad and Turney (2012). It contains more than 10000 words for a set of discrete emotion classes. Other resources include WordNet Affect (Strapparava and Valitutti, 2004) which distinguishes particular word classes. Further, annotated corpora have been created for a set of different domains, for instance fairy tales (Alm et al., 2005), Blogs (Aman and Szpakowicz, 2007), Twitter (Mohammad et al., 2017; Schuff et al., 2017; Mohammad, 2012; Mohammad and Bravo-Marquez, 2017a; Klinger et al., 2018), Facebook (Preořuc-Pietro et al., 2016), news headlines (Strapparava and Mihalcea, 2007), dialogues (Li et al., 2017), literature (Kim et al., 2017), or self reports on emotion events (Scherer, 1997) (see (Bostan and Klinger, 2018) for an overview).

To automatically assign emotions to textual units, the application of dictionaries has been a popular approach and still is, particularly in domains without annotated corpora. Another approach to overcome the lack of huge amounts of annotated training data in a particular domain or for a specific topic is to exploit distant supervision: use the signal of occurrences of emoticons or specific hashtags or words to automatically label the data. This is sometimes referred to as self-labeling (Klinger et al., 2018; Pool and Nissim, 2016; Felbo et al., 2017; Wang et al., 2012).



Figure 1: The setup of the driving simulator.

A variety of classification approaches have been tested, including SNoW (Alm et al., 2005), support vector machines (Aman and Szpakowicz, 2007), maximum entropy classification, long short-term memory network, and convolutional neural network models (Schuff et al., 2017, *i.a.*). More recently, the state of the art is the use of transfer learning from noisy annotations to more specific predictions (Felbo et al., 2017). Still, it has been shown that transferring from one domain to another is challenging, as the way emotions are expressed varies between areas (Bostan and Klinger, 2018). The approach by Felbo et al. (2017) is different to our work as they use a huge noisy data set for pre-training the model while we use small high quality data sets instead.

Recently, the state of the art has also been pushed forward with a set of shared tasks, in which the participants with top results mostly exploit deep learning methods for prediction based on pretrained structures like embeddings or language models (Klinger et al., 2018; Mohammad et al., 2018; Mohammad and Bravo-Marquez, 2017a).

Our work follows this approach and builds up on embeddings with deep learning. Furthermore, we approach the application and adaption of text-based classifiers to the automotive domain with transfer learning.

3 Data set Collection

The first contribution of this paper is the construction of the AMMER data set which we describe in the following. We focus on the drivers' interactions with both a virtual agent as well as a co-driver. To collect the data in a safe and controlled environment and to be able to consider a variety of predefined driving situations, the study was conducted in a driving simulator.

Type	Example
D–A, beginning	Wie geht es dir gerade und wie sind deine Gedanken zur bevorstehenden Fahrt? <i>How are you doing right now? What are your thoughts about the upcoming drive?</i>
D–A, reaching destination	Bei über 50 Teilnehmern hast du die zweitschnellste Zeit erreicht. Was glaubst du? Wie hast du es geschafft so schnell zu sein? <i>Among more than 50 participants you achieved the second best result. What do you think? How did you manage to achieve that?</i>
D–A, after driving	Du hast im letzten Streckenabschnitt ein paar Mal stark gebremst. Was ist da passiert? <i>In the last section, you slowed down multiple times. What happened?</i>
D–Co, low-demand section	Erinnern Sie sich an Ihren letzten Urlaub. Bitte beschreiben Sie, wie dieser Urlaub für Sie war? <i>Remember your last vacation. Please describe how it was.</i>

Table 1: Examples for triggered interactions with translations to English. (D: Driver, A: Agent, Co: Co-Driver)

3.1 Study Setup and Design

The study environment consists of a fixed-base driving simulator running Vires’s VTD (Virtual Test Drive, v2.2.0) simulation software (<https://vires.com/vtd-vires-virtual-test-drive/>). The vehicle has an automatic transmission, a steering wheel and gas and brake pedals. We collect data from video, speech and biosignals (Empatica E4 to record heart rate, electrodermal activity, skin temperature, not further used in this paper) and questionnaires. Two RGB cameras are fixed in the vehicle to capture the drivers face, one at the sun shield above the drivers seat and one in the middle of the dashboard. A microphone is placed on the center console. One experimenter sits next to the driver, the other behind the simulator. The virtual agent accompanying the drive is realized as Wizard-of-Oz prototype which enables the experimenter to manually trigger prerecorded voice samples playing trough the in-car speakers and to bring new content to the center screen. Figure 1 shows the driving simulator.

The experimental setting is comparable to an

everyday driving task. Participants are told that the goal of the study is to evaluate and to improve an intelligent driving assistant. To increase the probability of emotions to arise, participants are instructed to reach the destination of the route as fast as possible while following traffic rules and speed limits. They are informed that the time needed for the task would be compared to other participants. The route comprises highways, rural roads, and city streets. A navigation system with voice commands and information on the screen keeps the participants on the predefined track.

To trigger emotion changes in the participant, we use the following events: (i) a car on the right lane cutting off to the left lane when participants try to overtake followed by trucks blocking both lanes with a slow overtaking maneuver (ii) a skateboarder who appears unexpectedly on the street and (iii) participants are praised for reaching the destination unexpectedly quickly in comparison to previous participants.

Based on these events, we trigger three interactions (Table 1 provides examples) with the intelligent agent (*Driver-Agent Interactions, D–A*). Pretending to be aware of the current situation, e.g., to recognize unusual driving behavior such as strong braking, the agent asks the driver to explain his subjective perception of these events in detail. Additionally, we trigger two more interactions with the intelligent agent at the beginning and at the end of the drive, where participants are asked to describe their mood and thoughts regarding the (upcoming) drive. This results in five interactions between the driver and the virtual agent.

Furthermore, the co-driver asks three different questions during sessions with light traffic and low cognitive demand (*Driver-Co-Driver Interactions, D–Co*). These questions are more general and non-traffic-related and aim at triggering the participants’ memory and fantasy. Participants are asked to describe their last vacation, their dream house and their idea of the perfect job. In sum, there are eight interactions per participant (5 D–A, 3 D–Co).

3.2 Procedure

At the beginning of the study, participants were welcomed and the upcoming study procedure was explained. Subsequently, participants signed a consent form and completed a questionnaire to provide demographic information. After that, the co-driving experimenter started with the instruction

E	IT	Example
J	A	Ich glaube, weil ich ziemlich schnell auf Situationen reagieren kann, weil ich eine ziemlich gute Reaktion habe. Und ich würde auch behaupten, dass ich relativ vorausschauend fahre, weil ich schon einiges an Fahrerfahrung mitbringe. <i>I think because I can respond to situations very quickly because my reaction is very good. And I would say that I drive foresightful because I have a lot of driving experience.</i>
J	C	Letzter Urlaub war im September 2018. Singapur und Bali. War sehr schön. Erholung, andere Kultur, andere Länder. War sehr gut und ist zu wiederholen. <i>Last vacation was in September 2018. Singapore and Bali. It was beautiful. Recreation, different culture, different countries. It was very good and needs repetition.</i>
A	A	Zwei bis drei Mal Fahrzeuge, die Kolonne fuhren. Und das letzte Fahrzeug hat, für mein Gefühl, sehr ruckartig und mit wenig nach hinten zu schauen, die Spur gewechselt und mich dazu gezwungen, dann doch noch meine Geschwindigkeit zu reduzieren. <i>Two or three times vehicles were driving behind each other. The last vehicle cut off my lane, in my opinion very quickly and without looking back and forced me to slow down.</i>
A	C	Mir geht es nicht besonders gut. Die Fahrt war sehr stressig. Ich schwitze ziemlich. <i>I'm not feeling well. The ride was stressful. I am sweating.</i>
I	A	Letzter Urlaub war nicht so gut für mich. Obwohl. Naja doch. Der letzte war schon wieder gut. Das war im Sommer. Da war es nämlich so abartig warm dieses Jahr. Und wir haben bei uns daheim. Also ich komme ja vom Land. Wir haben bei uns daheim auf dem Land unsern Wohnwagen ausgebaut. <i>Last vacation was not so good for me. Although. Well, yes. The last one was good. It was in summer. It was very warm this year. And we have at home. I come from the countryside. We have furnished our mobile home.</i>
I	C	Ein Mensch ist über die Straße gelaufen und ich habe ihn zuerst nicht gesehen. <i>A human crossed the street and I haven't seen him in the first moment.</i>
B	A	Ich habe mich immer an die Richtgeschwindigkeit gehalten. Und ja. Ich weiß auch nicht. <i>I always followed the recommended velocity. And, well. I don't know.</i>
B	C	Ja. Nicht viel arbeiten und viel Geld verdienen. <i>Yes. Not working much and earning a lot of money.</i>
R	A	Mir geht es gut und ich bin gespannt auf die Fahrt. Ich denke, es macht Spaß. <i>I am fine and I am looking forward to the ride. I think it will be fun.</i>
R	C	Ja, ich erinnere mich an den letzten Urlaub und der war schön, war erholsam und war warm. <i>Yes, I remember the last vacation. It was nice, recreative and warm.</i>
N	A	Es sind Autos von der rechten Spur auf meine Spur gezogen, welche davor deutlich langsamer waren. <i>Cars were changing into my lane, which were slower before.</i>
N	C	Ein Haus, das relativ alleine für sich steht. Am besten am Meer und mit einem grünen Garten. Und ja. Viel Platz für sich. <i>A house with space around. In the best case at the sea and with a green garden. And yes. A lot of space for us.</i>

Table 2: Examples from the collected data set (with translation to English). E: Emotion, IT: interaction type with agent (A) and with Codriver (C). J: Joy, A: Annoyance, I: Insecurity, B: Boredom, R: Relaxation, N: No emotion.

in the simulator which was followed by a familiarization drive consisting of highway and city driving and covering different driving maneuvers such as tight corners, lane changing and strong braking. Subsequently, participants started with the main driving task. The drive had a duration of 20 minutes containing the eight previously mentioned speech interactions. After the completion of the drive, the actual goal of improving automatic emotional recognition was revealed and a standard emotional intelligence questionnaire, namely the TEIQue-SF (Cooper and Petrides, 2010), was handed to the participants. Finally, a retrospective interview was conducted, in which participants were played recordings of their in-car interactions and asked to give discrete (annoyance, insecurity, joy, relaxation, boredom, none, following (Dittrich and Zepf, 2019)) as well as dimensional (valence, arousal, dominance (Posner et al., 2005) on a 11-

point scale) emotion ratings for the interactions and the according situations. We only use the discrete class annotations in this paper.

3.3 Data Analysis

Overall, 36 participants aged 18 to 64 years ($\mu=28.89$, $\sigma=12.58$) completed the experiment. This leads to 288 interactions, 180 between driver and the agent and 108 between driver and co-driver. The emotion self-ratings from the participants yielded 90 utterances labeled with *joy*, 26 with *annoyance*, 49 with *insecurity*, 9 with *boredom*, 111 with *relaxation* and 3 with *no emotion*. One example interaction per interaction type and emotion is shown in Table 2. For further experiments, we only use *joy*, *annoyance/anger*, and *insecurity/fear* due to the small sample size for *boredom* and *no emotion* and under the assumption that *relaxation* brings little expressivity.

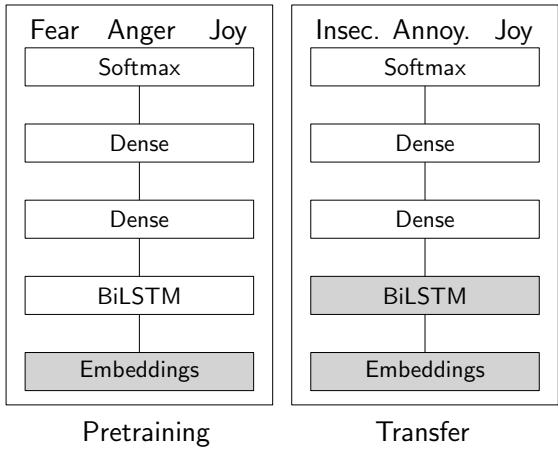


Figure 2: Model for Transfer Learning from Text. Grey boxes contain frozen parameters in the corresponding learning step.

4 Methods

4.1 Emotion Recognition from Facial Expressions

We preprocess the visual data by extracting the sequence of images for each interaction from the point where the agent’s or the co-driver’s question was completely uttered until the driver’s response stops. The average length is 16.3 seconds, with the minimum at 2.2s and the maximum at 54.7s. We apply an off-the-shelf tool for emotion recognition (the manufacturer cannot be disclosed due to licensing restrictions). It delivers frame-by-frame scores ($\in [0; 100]$) for discrete emotional states of *joy*, *anger* and *fear*. While *joy* corresponds directly to our annotation, we map *anger* to our label *annoyance* and *fear* to our label *insecurity*. The maximal average score across all frames constitutes the overall classification for the video sequence. Frames where the software is not able to detect the face are ignored.

4.2 Emotion Recognition from Audio Signal

We extract the audio signal for the same sequence as described for facial expressions and apply an off-the-shelf tool for emotion recognition. The software delivers single classification scores for a set of 24 discrete emotions for the entire utterance. We consider the outputs for the states of joy, anger, and fear, mapping analogously to our classes as for facial expressions. Low-confidence predictions are interpreted as “no emotion”. We accept the emotion with the highest score as the discrete prediction otherwise.

4.3 Emotion Recognition from Transcribed Utterances

For the emotion recognition from text, we manually transcribe all utterances of our AMMER study. To exploit existing and available data sets which are larger than the AMMER data set, we develop a transfer learning approach. We use a neural network with an embedding layer (frozen weights, pre-trained on Common Crawl and Wikipedia (Grave et al., 2018)), a bidirectional LSTM (Schuster and Paliwal, 1997), and two dense layers followed by a soft max output layer. This setup is inspired by (Andryushechkin et al., 2017). We use a dropout rate of 0.3 in all layers and optimize with Adam (Kingma and Ba, 2015) with a learning rate of 10^{-5} (These parameters are the same for all further experiments). We build on top of the Keras library with the TensorFlow backend. We consider this setup our *baseline model*.

We train models on a variety of corpora, namely the common format published by (Bostan and Klinger, 2018) of the *FigureEight* (formally known as Crowdflower) data set of social media, the *ISEAR* data (Scherer and Wallbott, 1994) (self-reported emotional events), and, the Twitter Emotion Corpus (TEC, weakly annotated Tweets with #anger, #disgust, #fear, #happy, #sadness, and #surprise, Mohammad (2012)). From all corpora, we use instances with labels *fear*, *anger*, or *joy*. These corpora are English, however, we do predictions on German utterances. Therefore, each corpus is preprocessed to German with Google Translate¹. We remove URLs, user tags (“@Username”), punctuation and hash signs. The distributions of the data sets are shown in Table 3.

To adapt models trained on these data, we apply *transfer learning* as follows: The model is first trained until convergence on one out-of-domain corpus (only on classes *fear*, *joy*, *anger* for compatibility reasons). Then, the parameters of the bi-LSTM layer are frozen and the remaining layers are further trained on AMMER. This procedure is illustrated in Figure 2

5 Results

5.1 Facial Expressions and Audio

Table 4 shows the confusion matrices for facial and audio emotion recognition on our complete AMMER data set and Table 5 shows the re-

¹<http://translate.google.com>, performed on January 4, 2019

Data set	Fear	Anger	Joy	Total
Figure8	8,419	1,419	9,179	19,017
EmoInt	2,252	1,701	1,616	5,569
ISEAR	1,095	1,096	1,094	3,285
TEC	2,782	1,534	8,132	12,448
AMMER	49	26	90	165

Table 3: Class distribution of the used data sets for the considered emotional states (Figure8 (Figure Eight, 2016), EmoInt (Mohammad and Bravo-Marquez, 2017b), ISEAR, (Scherer, 1997), TEC (Mohammad, 2012), AMMER (this paper)).

		Vision			Total
	Fear	Anger	Joy		
Insecurity	11	17	21		49
Annoyance	10	7	9		26
Joy	24	27	39		90
Total	45	51	69		165
		Audio			Total
	Fear	Anger	Joy	No	
Insecurity	17	14	1	17	49
Annoyance	12	7	0	7	26
Joy	27	26	4	33	90
Total	56	47	5	57	165
		Transfer Learning Text			Total
	Fear	Anger	Joy	No	
Insecurity	33	0	16		49
Annoyance	7	4	15		26
Joy	1	1	88		90
Total	41	5	119		165

Table 4: Confusion Matrix for Face Classification and Audio Classification (on full AMMER data) and for transfer learning from text (training set of EmoInt and test set of AMMER). Insecurity, annoyance and joy are the gold labels. Fear, anger and joy are predictions.

sults per class for each method, including facial and audio data and micro and macro averages. The classification from facial expressions yields a macro-averaged F_1 score of 33 % across the three emotions joy, insecurity, and annoyance ($P=0.31$, $R=0.35$). While the classification results for joy

are promising ($R=43\%$, $P=57\%$), the distinction of insecurity and annoyance from the other classes appears to be more challenging.

Regarding the audio signal, we observe a macro F_1 score of 29 % ($P=42\%$, $R=22\%$). There is a bias towards negative emotions, which results in a small number of detected joy predictions ($R=4\%$). Insecurity and annoyance are frequently confused.

5.2 Text from Transcribed Utterances

The experimental setting for the evaluation of emotion recognition from text is as follows: We evaluate the BiLSTM model in three different experiments: (1) in-domain, (2) out-of-domain and (3) transfer learning. For all experiments we train on the classes *anger/annoyance*, *fear/insecurity* and *joy*. Table 6 shows all results for the comparison of these experimental settings.

5.2.1 Experiment 1: In-Domain application

We first set a baseline by validating our models on established corpora. We train the baseline model on 60 % of each data set listed in Table 3 and evaluate that model with 40 % of the data from the same domain (results shown in the column “In-Domain” in Table 6). Excluding AMMER, we achieve an average micro F_1 of 68 %, with best results of $F_1=73\%$ on TEC. The model trained on our AMMER corpus achieves an F_1 score of 57%. This is most probably due to the small size of this data set and the class bias towards *joy*, which makes up more than half of the data set. These results are mostly in line with Bostan and Klinger (2018).

5.2.2 Experiment 2: Simple Out-Of-Domain application

Now we analyze how well the models trained in Experiment 1 perform when applied to our data set. The results are shown in column “Simple” in Table 6. We observe a clear drop in performance, with

	Vision			Audio			Text (TL)		
	P	R	F ₁	P	R	F ₁	P	R	F ₁
Insecurity	24	22	23	31	35	33	80	67	73
Annoyance	14	39	21	15	27	19	80	15	26
Joy	57	43	49	80	4	8	74	98	84
Macro-avg	32	35	33	42	22	29	78	60	68
Micro-avg	34	34	34	26	17	21	76	76	76

Table 5: Performance for classification from vision, audio, and transfer learning from text (training set of EmoInt).

Train Corpus	In-Domain	Out-of-domain			Transfer L.
		Simple	Joint C.	Transfer L.	
Figure8	66	55	59	76	
EmoInt	62	48	56	76	
TEC	73	55	58	76	
ISEAR	70	35	59	72	
AMMER	57	—	—	—	

Table 6: Results in micro F₁ for Experiment 1 (in-domain), Experiment 2 and 3 (out-of-domain with and without transfer learning).

an average of F₁=48 %. The best performing model is again the one trained on TEC, en par with the one trained on the Figure8 data. The model trained on ISEAR performs second best in Experiment 1, it performs worst in Experiment 2.

5.2.3 Experiment 3: Transfer Learning application

To adapt models trained on previously existing data sets to our particular application, the AMMER corpus, we apply transfer learning. Here, we perform leave-one-out cross validation. As pre-trained models we use each model from Experiment 1 and further optimize with the training subset of each crossvalidation iteration of AMMER. The results are shown in the column “Transfer L.” in Table 6. The confusion matrix is also depicted in Table 4.

With this procedure we achieve an average performance of F₁=75 %, being better than the results from the in-domain Experiment 1. The best performance of F₁=76 % is achieved with the model pre-trained on each data set, except for ISEAR. All transfer learning models clearly outperform their

simple out-of-domain counterpart.

To ensure that this performance increase is not only due to the larger data set, we compare these results to training the model without transfer on a corpus consisting of each corpus together with AMMER (again, in leave-one-out crossvalidation). These results are depicted in column “Joint C.”. Thus, both settings, “transfer learning” and “joint corpus” have access to the same information.

The results show an increase in performance in contrast to not using AMMER for training, however, the transfer approach based on partial retraining the model shows a clear improvement for all models (by 7pp for Figure8, 10pp for EmoInt, 8pp for TEC, 13pp for ISEAR) compared to the ”Joint” setup.

6 Summary & Future Work

We described the creation of the multimodal AMMER data with emotional speech interactions between a driver and both a virtual agent and a co-driver. We analyzed the modalities of facial expressions, acoustics, and transcribed utterances regarding their potential for emotion recognition during in-car speech interactions. We applied off-the-shelf emotion recognition tools for facial expressions and acoustics. For transcribed text, we developed a neural network-based classifier with transfer learning exploiting existing annotated corpora. We find that analyzing transcribed utterances is most promising for classification of the three emotional states of joy, annoyance and insecurity.

Our results for facial expressions indicate that there is potential for the classification of joy, however, the states of annoyance and insecurity are not well recognized. Future work needs to investigate more sophisticated approaches to map frame predictions to sequence predictions. Furthermore, movements of the mouth region during speech inter-

actions might negatively influence the classification from facial expressions. Therefore, the question remains how facial expressions can best contribute to multimodal detection in speech interactions.

Regarding the classification from the acoustic signal, the application of off-the-shelf classifiers without further adjustments seems to be challenging. We find a strong bias towards negative emotional states for our experimental setting. For instance, the personalization of the recognition algorithm (*e.g.*, mean and standard deviation normalization) could help to adapt the classification for specific speakers and thus to reduce this bias. Further, the acoustic environment in the vehicle interior has special properties and the recognition software might need further adaptations.

Our transfer learning-based text classifier shows considerably better results. This is a substantial result in its own, as only one previous method for transfer learning in emotion recognition has been proposed, in which a sentiment/emotion specific source for labels in pre-training has been used, to the best of our knowledge (Felbo et al., 2017). Other applications of transfer learning from general language models include (Rozental et al., 2018; Chronopoulou et al., 2018, *i.a.*). Our approach is substantially different, not being trained on a huge amount of noisy data, but on smaller out-of-domain sets of higher quality. This result suggests that emotion classification systems which work across domains can be developed with reasonable effort.

For a productive application of emotion detection in the context of speech events we conclude that a deployed system might perform best with a speech-to-text module followed by an analysis of the text. Further, in this work, we did not explore an ensemble model or the interaction of different modalities. Thus, future work should investigate the fusion of multiple modalities in a single classifier.

Acknowledgment

We thank Laura-Ana-Maria Bostan for discussions and data set preparations. This research has partially been funded by the German Research Council (DFG), project SEAT (KL 2869/1-1).

References

- Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: Machine learning for text-based emotion prediction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 579–586, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.
- Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In Václav Matoušek and Pavel Mautner, editors, *Text, Speech and Dialogue*, pages 196–205, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Christos-Nikolaos Anagnostopoulos, Theodoros Iliou, and Ioannis Giannoukos. 2015. Features and classifiers for emotion recognition from speech: a survey from 2000 to 2011. *Artificial Intelligence Review*, 43(2).
- Vladimir Andryushchkin, Ian Wood, and James O’Neill. 2017. NUIG at EmoInt-2017: BiLSTM and SVR ensemble to detect emotion intensity. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 175–179, Copenhagen, Denmark, September. Association for Computational Linguistics.
- Hynek Boril, Omid Sadjadi, and John H. L. Hansen Hansen. 2011. UTDrive: emotion and cognitive load classification for in-vehicle scenarios. In *th Biennial Workshop on Digital Signal Processing for In-Vehicle Systems, DSP 2011*.
- Laura-Ana-Maria Bostan and Roman Klinger. 2018. An analysis of annotated corpora for emotion classification in text. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2104–2119, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.
- Scott Brave, Clifford Nass, and Kevin Hutchinson. 2005. Computers that care: investigating the effects of orientation of emotion exhibited by an embodied computer agent. *International journal of human-computer studies*, 62(2).
- Carlos Busso, Zhigang Deng, Serdar Yildirim, Murtaza Bulut, Chul Min Lee, Abe Kazemzadeh, Sungbok Lee, Ulrich Neumann, and Shrikanth Narayanan. 2004. Analysis of emotion recognition using facial expressions, speech and multimodal information. In *Proceedings of the 6th International Conference on Multimodal Interfaces, ICMI ’04*, pages 205–211, New York, NY, USA. ACM.
- Alexandra Chronopoulou, Aikaterini Margatina, Christos Baziotis, and Alexandros Potamianos. 2018. NTUA-SLP at IEST 2018: Ensemble of neural transfer methods for implicit emotion classification. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 57–64, Brussels, Belgium, October. Association for Computational Linguistics.

- Andrew Cooper and Konstantinos Vassilis Petrides. 2010. A psychometric analysis of the trait emotional intelligence questionnaire—short form (TEIQue-SF) using item response theory. *Journal of personality assessment*, 92(5):449–457.
- Amy Coplan and Peter Goldie. 2011. *Empathy: Philosophical and psychological perspectives*. Oxford University Press.
- Monique Dittrich and Sebastian Zepf. 2019. Exploring the validity of methods to track emotions behind the wheel. In Harri Oinas-Kukkonen, Khin Than Win, Evangelos Karapanos, Pasi Karppinen, and Eleni Kyza, editors, *Persuasive Technology: Development of Persuasive and Behavior Change Support Systems*, pages 115–127, Cham. Springer International Publishing.
- Paul Ekman and Wallace V. Friesen. 1978. Facial action coding system: Investigator’s guide. *Consulting Psychologists Press*.
- Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion*, 6.
- Moataz El Ayadi, Mohamed S Kamel, and Fakhri Karray. 2011. Survey on speech emotion recognition: Features, classification schemes, and databases. *Pattern Recognition*, 44(3).
- Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625, Copenhagen, Denmark, September. Association for Computational Linguistics.
- Figure Eight. 2016. Sentiment analysis: Emotion in text. Online. <https://www.figure-eight.com/data/sentiment-analysis-emotion-text/>.
- Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan, May. European Languages Resources Association (ELRA).
- Helen Harris and Clifford Nass. 2011. Emotion regulation for frustrating driving contexts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ’11*, pages 749–752.
- Klas Ihme, Christina Dömeland, Maria Freese, and Meike Jipp. 2018. Frustration in the Face of the Driver: A Simulator Study on Facial Muscle Activity during Frustrated Driving. *Interaction Studies*, 19.
- Heechul Jung, Sihaeng Lee, Junho Yim, Sunjeong Park, and Junmo Kim. 2015. Joint fine-tuning in deep neural networks for facial expression recognition. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2983–2991.
- Bo-Kyeong Kim, Jihyeon Roh, Suh-Yeon Dong, and Soo-Young Lee. 2016. Hierarchical committee of deep convolutional neural networks for robust facial expression recognition. *Journal on Multimodal User Interfaces*, 10(2).
- Evgeny Kim, Sebastian Padó, and Roman Klinger. 2017. Investigating the relationship between literary genres and emotional plot development. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 17–26, Vancouver, Canada, August. Association for Computational Linguistics.
- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.
- Jonathan Klein, Youngme Moon, and Rosalind W. Picard. 2002. This computer responds to user frustration: Theory, design, and results. *Interacting with computers*, 14(2).
- Roman Klinger, Orphée De Clercq, Saif Mohammad, and Alexandra Balahur. 2018. IEST: WASSA-2018 implicit emotions shared task. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 31–42, Brussels, Belgium, October. Association for Computational Linguistics.
- Jinkyu Lee and Ivan Tashev. 2015. High-level feature representation using recurrent neural network for speech emotion recognition. In *Interspeech*. ISCA – International Speech Communication Association.
- Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan, November. Asian Federation of Natural Language Processing.
- James J. Lien, Takeo Kanade, Jeffrey F. Cohn, and Ching-Chung Li. 1998. Automated facial expression recognition based on facs action units. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 390–395.
- Zhiyi Ma, Marwa Mahmoud, Peter Robinson, Eduardo Dias, and Lee Skrypchuk. 2017. Automatic detection of a driver’s complex mental states. In Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Giuseppe Borruso, Carmelo M. Torre, Ana Maria A.C. Rocha, David Taniar, Bernady O. Apduhan, Elena Stankova, and Alfredo Cuzzocrea, editors, *Computational Science and Its Applications – ICCSA 2017*, pages 678–691, Cham. Springer International Publishing.

- Judi Mesman, Harriet Oster, and Linda Camras. 2012. Parental sensitivity to infant distress: what do discrete negative emotions have to do with it? *Attachment & Human Development*, 14(4).
- Saif Mohammad and Felipe Bravo-Marquez. 2017a. Emotion intensities in tweets. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 65–77, Vancouver, Canada, August. Association for Computational Linguistics.
- Saif Mohammad and Felipe Bravo-Marquez. 2017b. Emotion intensities in tweets. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 65–77, Vancouver, Canada, August. Association for Computational Linguistics.
- Saif M Mohammad and Peter D Turney. 2012. Crowd-sourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3).
- Saif M. Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and sentiment in tweets. *ACM Trans. Internet Technol.*, 17(3).
- Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. SemEval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17, New Orleans, Louisiana, June. Association for Computational Linguistics.
- Saif Mohammad. 2012. #emotional tweets. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 246–255, Montréal, Canada. Association for Computational Linguistics.
- Clifford Nass, Ing-Marie Jonsson, Helen Harris, Ben Reaves, Jack Endo, Scott Brave, and Leila Takayama. 2005. Improving automotive safety by pairing driver emotion and car voice emotion. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 1973–1976.
- Michael Neumann and Ngoc Thang Vu. 2017. Attentive convolutional neural network based speech emotion recognition: A study on the impact of input features, signal length, and acted speech. In *Interspeech*. ISCA – International Speech Communication Association.
- Dimitri Palaz, Mathew Magimai-Doss, and Ronan Collobert. 2015. Analysis of cnn-based speech recognition system using raw speech as input. In *Interspeech*. ISCA – International Speech Communication Association.
- Maja Pantic, Nicu Sebe, Jeffrey F. Cohn, and Thomas Huang. 2005. Affective multimodal human-computer interaction. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05, pages 669–676.
- Timo Partala and Veikko Surakka. 2004. The effects of affective interventions in human–computer interaction. *Interacting with computers*, 16(2).
- Maurizio Paschero, G. Del Vescovo, L. Benucci, Antonello Rizzi, Marco Santello, Gianluca Fabbri, and F. M. Frattale Mascioli. 2012. A real time classifier for emotion and stress recognition in a vehicle driver. In *2012 IEEE International Symposium on Industrial Electronics*, pages 1690–1695.
- James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of LIWC2015.
- Robert Plutchik. 1980. A general psychoevolutionary theory of emotion. *Theories of emotion*, 1.
- Chris Pool and Malvina Nissim. 2016. Distant supervision for emotion detection using Facebook reactions. In *Proceedings of the Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media (PEOPLES)*, pages 30–39, Osaka, Japan, December. The COLING 2016 Organizing Committee.
- Jonathan Posner, James A. Russell, and Bradley S. Peterson. 2005. The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. *Development and psychopathology*, 17(3):715–734.
- Daniel Preoțiuc-Pietro, H. Andrew Schwartz, Gregory Park, Johannes Eichstaedt, Margaret Kern, Lyle Ungar, and Elisabeth Shulman. 2016. Modelling valence and arousal in Facebook posts. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 9–15, San Diego, California, June. Association for Computational Linguistics.
- K. Sreenivasa Rao, Shashidhar G. Koolagudi, and Ramu Reddy Vempada. 2013. Emotion recognition from speech using global and local prosodic features. *International journal of speech technology*, 16(2).
- Alon Rozental, Daniel Fleischer, and Zohar Kelrich. 2018. Amobee at IEST 2018: Transfer learning from language models. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 43–49, Brussels, Belgium, October. Association for Computational Linguistics.
- Klaus R Scherer and Harald G. Wallbott. 1994. Evidence for universality and cultural variation of differential emotion response patterning. *Journal of personality and social psychology*, 66(2).

- Klaus R. Scherer. 1997. Profiles of emotion-antecedent appraisal: Testing theoretical predictions across cultures. *Cognition & Emotion*, 11(2).
- Hendrik Schuff, Jeremy Barnes, Julian Mohme, Sebastian Padó, and Roman Klinger. 2017. Annotation, modelling and analysis of fine-grained emotions on a stance and sentiment detection corpus. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 13–23, Copenhagen, Denmark, September. Association for Computational Linguistics.
- Björn W. Schuller, Manfred K. Lang, and Gerhard Rigoll. 2006. Recognition of Spontaneous Emotions by Speech within Automotive Environment. In *Tagungsband Fortschritte der Akustik – DAGA 2006*, pages 57–58.
- Björn W. Schuller. 2018. Speech emotion recognition: Two decades in a nutshell, benchmarks, and ongoing trends. *Communications of the ACM*, 61(5):90–99.
- Mike Schuster and Kuldeep K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11).
- Nicu Sebe, Ira Cohen, and Thomas S. Huang. 2005. *Handbook of Pattern Recognition and Computer Vision*, chapter Multimodal Emotion Recognition. World Scientific.
- Mohammad Soleymani, Maja Pantic, and Thierry Pun. 2012. Multimodal emotion recognition in response to videos. *IEEE Transactions on Affective Computing*, 3(2).
- Carlo Strapparava and Rada Mihalcea. 2007. SemEval-2007 task 14: Affective text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 70–74, Prague, Czech Republic, June. Association for Computational Linguistics.
- Carlo Strapparava and Alessandro Valitutti. 2004. WordNet affect: an affective extension of WordNet. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May. European Language Resources Association (ELRA).
- Sujono and Alexander A. S. Gunawan. 2015. Face expression detection on kinect using active appearance model and fuzzy logic. In Widodo Budiharto, editor, *Proceedings of the International Conference on Computer Science and Computational Intelligence (ICCSI 2015)*, volume 59 of *Procedia Computer Science*, pages 268–274. Elsevier.
- Tessa-Karina Tews, Michael Oehl, Felix W. Siebert, Rainer Höger, and Helmut Faasch. 2011. Emotional human-machine interaction: Cues from facial expressions. In Michael J. Smith and Gavriel Salvendy, editors, *Human Interface and the Management of Information. Interacting with Information*, pages 641–650. Springer Berlin Heidelberg.
- George Trigeorgis, Fabien Ringeval, Raymond Brueckner, Erik Marchi, Mihalis A. Nicolaou, Björn Schuller, and Stefanos Zafeiriou. 2016. Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5200–5204.
- Dimitrios Ververidis, Constantine Kotropoulos, and Ioannis Pitas. 2004. Automatic emotional speech classification. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages I–593–I–596.
- Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P. Sheth. 2012. Harnessing twitter "big data" for automatic emotion identification. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Conference on Social Computing*, pages 587–592.

Visualising and evaluating the effects of combining active learning with word embedding features

Maria Skeppstedt¹, Rafal Rzepka^{2,3}, Kenji Araki², Andreas Kerren⁴

¹The Language Council of Sweden, the Institute for Language and Folklore, Sweden

²Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan

³RIKEN Center for Advanced Intelligence Project (AIP), Tokyo, Japan

⁴Department of Computer Science and Media Technology, Linnaeus University, Växjö, Sweden

maria.skeppstedt@sprakochfolkminnen.se, andreas.kerren@lnu.se,
{rzepka, araki}@ist.hokudai.ac.jp

Abstract

A tool that enables the use of active learning, as well as the incorporation of word embeddings, was evaluated for its ability to decrease the training data set size required for a named entity recognition model. Uncertainty-based active learning and the use of word embeddings led to very large performance improvements on small data sets for the entity categories PERSON and LOCATION. In contrast, the embedding features used were shown to be unsuitable for detecting entities belonging to the ORGANISATION category. The tool was also extended with functionality for visualising the usefulness of the active learning process and of the word embeddings used. The visualisations provided were able to indicate the performance differences between the entities, as well as differences with regards to usefulness of the embedding features.

1 Introduction

To acquire large training data sets by the use of low-cost crowdsourcing is not a universal solution for all annotation tasks. The ethical aspect could be one concern, as the concept of low-cost crowd annotations implies low-paid annotators (Martin et al., 2017). Other obstacles might be data privacy restrictions (e.g., when annotating clinical health records), or a lack of specialised competence among crowd workers, e.g., competence in the annotation task or in a specific language. Strategies for facilitating annotation are therefore important, also in the age of crowdsourcing.

A possible strategy for facilitating annotation is to minimise the amount of manually annotated data required, e.g., data required for the task of training a machine learning model. This could be achieved by (i) using active learning to actively select training samples useful to the model and (ii) training

the model on information that has been derived in an unsupervised fashion. There is a large body of research that has shown the effectiveness of using each one of these strategies individually, and there are also annotation tools/annotation tool extensions that incorporate these two strategies (Skeppstedt et al., 2016; Kucher et al., 2017). However, to the best of our knowledge, there are no studies that evaluate the effectiveness of this combined data reducing strategy provided by the tools. The first aim of this study is therefore to evaluate the effectiveness of one such tool, i.e., to evaluate whether using the tool leads to the expected decrease in data size required to train a machine learning model.

Also the annotation of a smaller data set can however be a time-consuming, and potentially boring, task. Gamification of the task is one previously explored strategy for solving this problem (Dumitrache et al., 2013; Venhuizen et al., 2013).

Another potential strategy for increasing the intrinsic motivation for the annotation task, is to make the annotator aware of the usefulness of the data that is being annotated. The second aim of this study is to take a first step towards exploring this strategy in the context of an active learning process. We aim to provide a suggestion for a visualisation of how the increasingly larger training data set, which results from the manual annotation effort, changes the model that is trained on this annotated data set. That is, a visualisation that has the potential to increase the human understanding of the active learning-based annotation process.

2 Background

The tool whose performance we have evaluated, and whose active learning process we have visualised, is the tool “PAL – a tool for Pre-annotation and Active Learning” (Skeppstedt et al., 2016). PAL is meant to be used as an extension to another annotation tool, e.g., BRAT (Stenetorp et al., 2012), for annotating data to be used for training

a named entity recognition (NER) model. While high performance is often reported for the NER task, e.g., for newswire texts (Sang and Meulder, 2003), the task is more difficult for noisy texts and when small training data sets are used. For instance, the best system on the ACL 2015 Workshop on Noisy User-generated Text achieved an F-score of 0.74 for PERSON, 0.50 for COMPANY, and 0.66 for GEO-LOCATION when using a training set of 2,950 tweets (Baldwin et al., 2015; Yamada et al., 2015).

PAL provides functionality for active data selection, as well as for incorporating unsupervised data in the form of word embeddings when training the models that are used for active data selection. The tool also offers annotation support in the form of pre-annotations. This is achieved by repeatedly retraining a NER model on the data that the annotator produces in BRAT and on information incorporated from word embeddings. The trained model can then be used for two purposes: (i) to actively import new annotation data into BRAT, i.e., to actively select data useful for improving the model, and (ii) to simplify the annotation by providing the annotator with pre-annotations in BRAT format. To allow the annotator to add, delete or change the span length of pre-annotated entities — instead of annotating from scratch — has been shown to reduce annotation time (Lingren et al., 2014).

PAL could, for instance, be used according to the annotation process suggested by Olsson (2008). That is, to first annotate an actively selected subset of a corpus to achieve a model that can perform pre-annotations with acceptable accuracy, and thereafter use this model for providing the annotator with pre-annotations when a larger corpus is annotated. Such a corpus might, for instance, be used for training a model that requires a large training data set to perform well. The current study focuses on the first part of such a use case, that is on the process of actively selecting training samples to achieve a model that recognises named entities with acceptable performance.

2.1 Approaches for minimising training data

To use active learning, instead of a random sampling of training data, has led to a reduction of the number of samples needed to train classifiers to recognise different entity types (Shen et al., 2004; Tomanek et al., 2007). The technique builds on the following idea: Data samples estimated to be

useful to a machine learning model are actively selected from a pool of unlabelled data. The selected samples are presented to an annotator for manual annotation, and the newly annotated data is then added to the set of labelled data that is available for training the model. This expanded training data set is then used to retrain the model, which in turn is applied in the next iteration in the process of actively selecting data. The estimate of a sample’s usefulness can, for instance, be based on the level of disagreement among different classifiers (Olsson, 2008, pp. 25–29), or on properties specific to the type of model used, e.g., a confidence measure provided by the model (Settles, 2009).

The other technique included in PAL for reducing the training data size is to incorporate features gathered in an unsupervised fashion, through the use of text distributional properties of word types. There is a large body of research that shows this technique to be effective for named entity recognition, e.g., the use of features in the form of Brown clusters (Miller et al., 2004) and more recently in the form of different types of word vectors automatically derived from large corpora (Sahlgren, 2006; Mikolov et al., 2013). Word vectors have for instance been incorporated in the feature set when using conditional random fields classifiers (Turian et al., 2010; Guo et al., 2014; Henriksson, 2015; Copara et al., 2016), or used as input to different types of neural network-based classifiers (Godin et al., 2015; dos Santos and Guimarães, 2015; Yang et al., 2016; Lample et al., 2016; Reimers and Gurevych, 2017). There is, however, less research that investigates the effects of using the two strategies of unsupervised features and active learning in tandem; in particular their effects on small data sets, i.e., the use case that we explore here.

2.2 Functionality of PAL

Each iteration in PAL is run in two steps. First, data positioned in PAL’s “folder for labelled data” is used for training a machine learning model; a model which is then used for selecting new data samples from PAL’s “folder for unlabelled data.” The model also provides BRAT-format pre-annotations for the selected data, enabling it to be directly imported into BRAT (Figure 3b). In the second step, which takes place after the data has been manually annotated, the data annotated in BRAT is moved into PAL’s “folder for labelled data”, to enable the next active learning iteration.

A basic feature vector for training the model, x_n , is constructed through representing each token by a concatenation of (i) the one-hot encoding for the token with (ii) the one-hot encoding for a configurable number of neighbours to the token.

The functionality of incorporating features derived in an unsupervised fashion is provided in PAL through an extension of the basic vector by a vector derived from pre-trained word embeddings. This is achieved by concatenating the basic feature vector with the word embedding vector that represents the token, as well as with the word embedding vectors that represent the neighbours of the token.

Information from gazetteers or information on which words were capitalised were not included in the feature set, to focus the experiment on the effects of the different strategies compared. This also makes the results somewhat more generalisable, e.g., to entity types that are not typically capitalised or for which gazetteers do not typically exist, or to languages that do not use an initial capital letter as a signal for names.

With the focus on making the data selection and model training process as comprehensible as possible for a human, we used the main classification method included in PAL, which is a token-level logistic regression classifier. That is, a classifier for which a human-interpretable confidence measure can be returned for each token in the pool of unlabelled data. The output of this unstructured predictor, is then post-processed into B/I-labels for tokens classified as an entity.

The confidence is then used for carrying out *uncertainty sampling* from the pool of unlabelled data (Settles, 2009). More specifically, the measure used is the difference in certainty level between the two most probable classifications for each of the tokens in the data pool. Given c_{p1} as the most probable classification and c_{p2} as the second most probable classification for the observation x_n , the uncertainty measure would be:

$$M_n = P(c_{p1}|x_n) - P(c_{p2}|x_n) \quad (1)$$

The smaller M_n , the higher is the uncertainty of the classifier and the higher is the sample ranked in the active selection process (Schein and Ungar, 2007).

PAL represents each training sample by the lowest M among the tokens it includes. For each iteration in the active selection process, samples that contain tokens with the lowest M -values are thereby selected. To achieve a variation among the

samples selected, PAL also imposes the constraint of not allowing the selected texts to include the same word twice, if this word is predicted by the model to be included in a named entity.

PAL accesses embeddings through Gensim (Řehůřek and Sojka, 2010) and uses Scikit-learn’s (Pedregosa et al., 2011) logistic regression classifier with a regularisation strength determined through cross-fold validation.

3 Method

The evaluation of PAL was carried out using the Broad Twitter Corpus (Derczynski et al., 2016), which consists of English tweets annotated for the three entities PERSON, LOCATION, and ORGANISATION. The corpus is sampled across different regions, temporal periods, and from different types of Twitter users, to ensure a large diversity of the entities included. Each of the three entity types was annotated separately.

We removed metadata in the form of hashtags and usernames starting with @, to make the task more similar to most previous NER tasks, where entities are mentioned in a textual context. The corpus is divided into six segments, each of them with a different signifying property, e.g., tweets from popular individuals, tweets from mainstream news, or tweets focused on one specific event. For performing the experiments we, however, sampled randomly from the corpus (as described below), without taking this structure into account.

3.1 Simulation of active learning

The active learning process in PAL was used in simulated mode as follows: the machine learning model was first trained on a small labelled data set consisting of 200 randomly selected tweets, i.e., a set representing an initial seed set. The task of the active learning algorithm would then be to select the most informative data points from the pool of unlabelled data. In the experiment, the “pool of unlabelled data” was simulated by the texts from the pre-labelled tweets in the Broad Twitter Corpus, and the corpus labels were used to simulate input in the form of manual annotations performed by the annotator.

For the experiment performed, we selected 20 tweets in each iteration. These 20 tweets and their corresponding labels were thus added to the set of labelled data, to simulate the process of them being manually annotated. The model was, thereafter,

retrained, and a new iteration in the process of actively selecting tweets was then carried out, until the set of labelled data contained 1,000 tweets.

A context window of the two most immediate neighbours was used, with a frequency cut-off of three occurrences for a neighbour to be included. Word embeddings from a word2vec skip-gram model, which had been pre-trained by Godin et al. (2015) on 400 million tweets, were used as unsupervised features.

3.2 Evaluating the active learning simulation

The strategies used in PAL for decreasing the training data size required were compared to a baseline strategy. A total of four different strategies were thus evaluated for their performance on a small training data set: (i) the baseline, with *random* data selection and a *basic feature* vector, (ii) data selection through *active learning* and the *basic feature* vector, (iii) *random* data selection and the feature vector extended with *word2vec features*, and finally (iv) data selection through *active learning* and the feature vector extended with *word2vec features*.

4,000 tweets were randomly selected from the Broad Twitter Corpus to simulate the pool of unlabelled data, and 2,000 other tweets were randomly selected to be used as evaluation data. From the simulated pool of data were then 200 tweets randomly selected to form the seed set.

Starting with this seed set, an evaluation was carried out of the four different strategies investigated. For one of the active learning strategies, the basic feature vector was used, and for the other, the word2vec extension. For every step in the iteration, the performance of the model was evaluated against the 2,000 tweets that formed the evaluation data, i.e., after 20 new training data samples had been actively added to the training data set.

For the two strategies that did not include active learning, each iteration instead consisted of a random selection of 20 new tweets from the simulated data pool. A new model was trained on data including these newly selected tweets, and then evaluated against the 2,000 tweets in the evaluation set. The same randomly selected data sets were used both for the setting with word2vec features and the setting without these features.

As results of the study were heavily dependent on the random selection of a number of small data sets, it was particularly important to make sure that results achieved were not due to chance. The entire

experiment was therefore repeated 10 times, each time with a new random selection of data pool, evaluation and seed set, as well as training data for the strategies not using active data selection. A separate experiment was carried out for each one of the three entity types LOCATION, ORGANISATION and PERSON, i.e., matching the manner in which the evaluation corpus had been annotated. Entities were represented by the BIO-encoding, and the classifications were evaluated using the CoNLL 2000 NER script (Tjong Kim Sang and Buchholz, 2000).

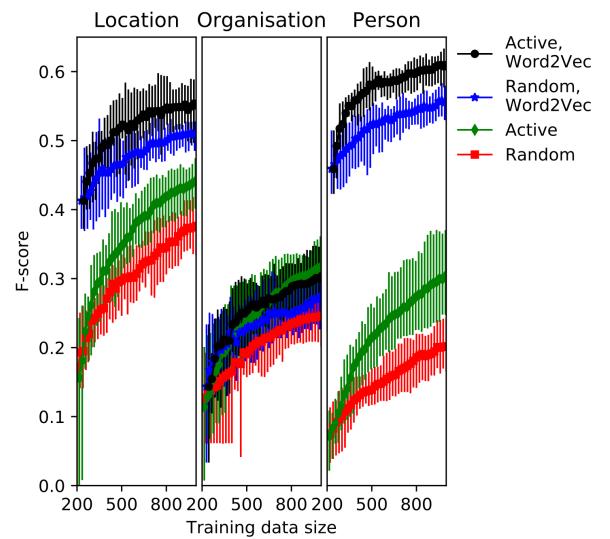


Figure 1: Average F-score for the ten experiment runs. The error bars show the interval between the minimum and maximum of the F-scores measured, and the x-axes show the number of training samples.

3.3 Visualising the active learning process

We extended PAL by enabling it to record statistics for the pool of unlabelled data for each iteration of active data selection. We also extended the tool by adding a command which allows the user to generate a visualisation of this recorded data. The visualisation aimed to increase the human understanding of the active learning and classifier training by (i) showing why a particular set of samples are chosen for manual annotation in each iteration, (ii) showing an indication of the usefulness of the embedding features used, through visualising how clusters formed by the embeddings correspond to the entity categories investigated, and (iii) showing how the classification uncertainty for the pool of unlabelled data changes when more data is annotated and used for training the model.

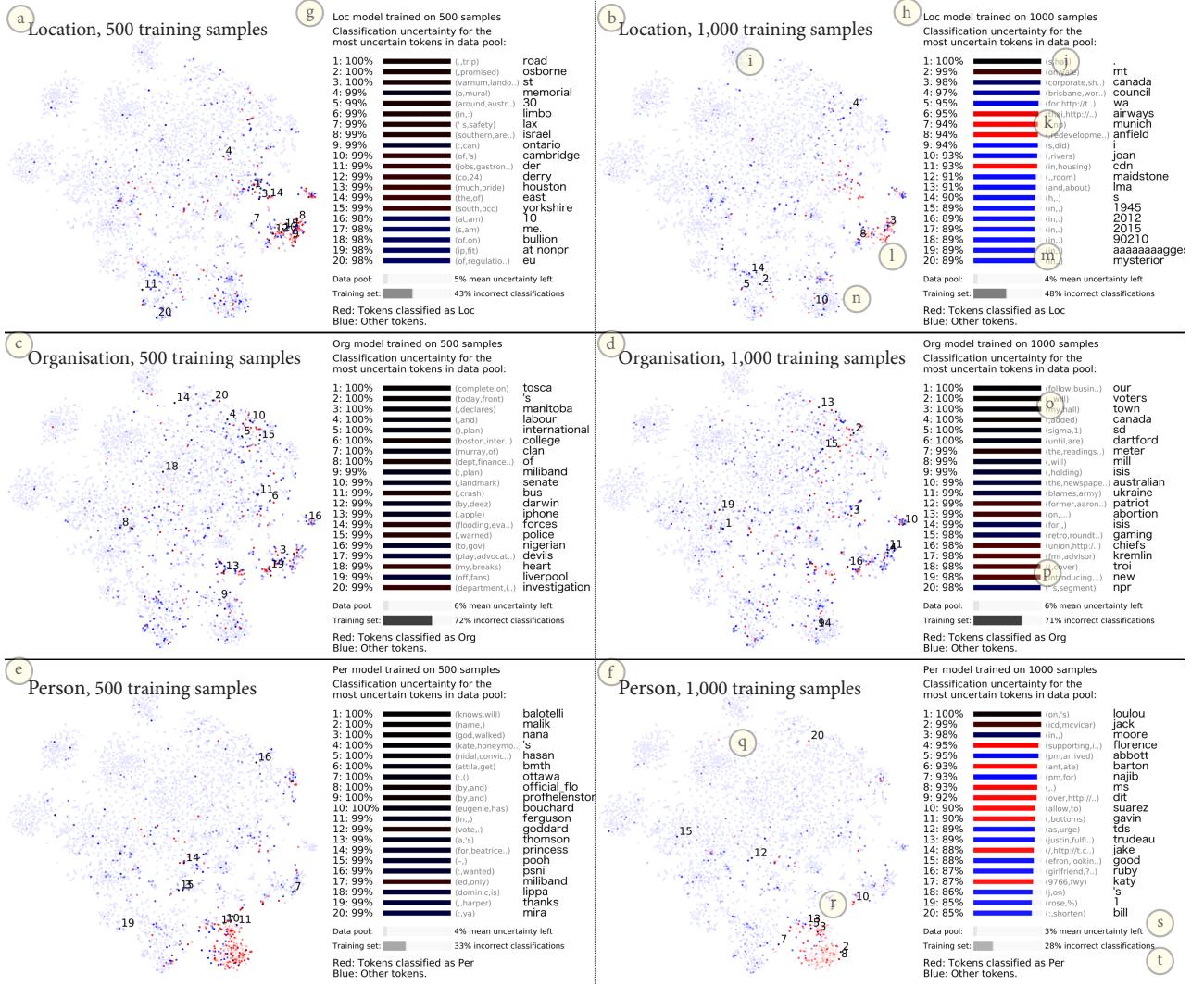


Figure 2: **(a-f)** Six subplots, two for each of the three entity categories. **(g-h)** The left-hand column: The model's uncertainty for classifying tokens in the pool of unlabelled data when 500 samples have been removed from the pool, labelled, and then used as training data for the model. **(i-l)** The right-hand column: Same as g, but with a training data size of 1,000 samples. **(m-n)** The colour red is used for signifying that a token has been classified by the model as belonging to the entity category in question (i.e., classified as a LOCATION, ORGANISATION or PERSON entity). **(o-p)** The colour blue is used for signifying that a token is not classified as belonging to the entity category in question. **(q-r)** The t-SNE plot and the bar chart use the same colour-coding for signifying the output of the machine learning model. The larger the uncertainty with which a token is classified by the model, the darker (i.e., closer to black) is the red or blue in which it is displayed. **(s-t)** Bar chart indicating mean model uncertainty for all words left in the pool of unlabelled data. **(t)** Bar chart indicating the proportion of incorrectly classified tokens when conducting cross-fold validation on the training set.

The advantage of applying the functionality in PAL that uses a token-level, logistic regression classifier for the data selection, and that selects samples

based on their most uncertain token, is that the selection process is easily explainable. That is, the first of the visualisation goals can be met by con-

veying a list of these tokens, for which the model was most uncertain, together with the model’s classification uncertainty for these tokens.

The second visualisation goal can be met by plotting a t-distributed stochastic neighbour embedding plot, t-SNE (van der Maaten and Hinton, 2008), of the word embeddings that were used as features. Plotted word embeddings can then be colour-coded according to how the word which they represent most often is classified. Thereby, a comparison between classifications by the trained model and clusters of word embeddings, as shown by the t-SNE plot, can be carried out.

To show the classification uncertainty of the most uncertain tokens also helps meeting the third visualisation goal. That is, changes in uncertainty for these most uncertain tokens indicate changes in model performance when the training data size increases. In addition, the colour-coding of the t-SNE plot can also be used for indicating whether the classification uncertainty for the tokens in the pool of unlabelled data changes when more data is labelled and used as training data.

3.4 Visualisations for another corpus

To verify that the visualisation also functions on another corpus than the English corpus that we used during development and for simulation of the process, we performed a small annotation experiment on a corpus of Japanese microblogs.¹

As white space is not normally used in Japanese text, we first performed a pre-processing using the MeCab tool (Kudo, 2006). That is, the text segments generated by MeCab was used, and white space was inserted between these segments. Thereby, the white space-based tokenisation included in Scikit-learn could be used as-is. As unsupervised features, we used word embedding vectors from a word2vec model that had been trained on Japanese texts, which had been segmented by MeCab and merged with the help of a dictionary².

For this corpus, we did not perform a simulation, but instead applied PAL for the authentic use case of annotating raw text data. That is, we used the facilities of active learning and pre-annotation that are available in PAL for annotating text, and gen-

¹<http://www.cs.cmu.edu/~lingwang/microtopia/#twittergold>
Microblogs collected with the criterium that they should contain the same content written in Japanese and in English (Ling et al., 2014), from which we used the Japanese parts.

²<https://github.com/shiroyagicorp/japanese-word2vec-model-builder>

erated a visualisation after each iteration. We imported the pre-annotations generated by PAL into the BRAT annotation tool, as shown in Figure 3, to modify or delete incorrect annotations and to add omitted ones. We used annotation guidelines for entity detection and tracking (EDT)³.

4 Results

Evaluation results in the form of an F-score measurement when evaluating against an external evaluation set are shown in Figure 1, while Figures 2 and 3 show the output of the proposed visualisations for the active learning process.

4.1 Evaluation results

The main lines in Figure 1 show the average F-scores for the ten re-runs for each training data size included in the experiment. The error bars show the minimum and maximum F-scores for the ten re-runs, i.e., giving an indication of the variation in the results achieved.

For the entity categories LOCATION and PERSON, average F-scores for the four different strategies produce four well-separated lines. Results are often separated, or close to separated, also when taking the lowest/highest value measured for the ten folds into account. Active data selection gives better results than random selection, and incorporating unsupervised features gives better results than not using them. The incorporation of unsupervised features is a more useful strategy than active data selection, and, more importantly, combining the two strategies is the overall most useful method.

Figure 1 further shows that while active learning was useful also for the category ORGANISATION, the use of word embeddings instead had a small negative impact on this category for a data set containing more than 600 samples.

4.2 Visualisation output

The visualisation functionality, with which we extended the PAL tool in this study, provides one visualisation of the unlabelled data pool for each iteration in the active learning process. The left-hand column in Figure 2 shows three visualisations, one for each of the three entity categories investigated. Each of them was generated in an active learning iteration when the training data set contained 500 samples. The right-hand column in the

³www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-edt-v4.2.6.pdf

figure shows visualisations for the three categories, that instead were generated when the training data set contained 1,000 data samples. All six subplots visualise the state of the pool when using active learning and the word2vec features.

Each subplot shows the state of the pool of unlabelled data. That is, each subplot contains an uncertainty colour-coded t-SNE visualisation of word embeddings that correspond to tokens present in the data pool, as well as a bar chart displaying the classification uncertainty for the 20 most uncertain tokens in the pool. Red colours in the t-SNE plot and the bar chart signify tokens that the model, trained on the currently available labelled data, classifies as belonging to the entity category in question, whereas blue colours indicate that this model classifies the token as outside of an entity. Darker colours in the t-SNE plot and the bar chart signify higher uncertainty for the classification.

In particular, the colours and lengths of the bars for PERSON and LOCATION show that there is a higher uncertainty for a model trained on 500 data samples than for a model trained on 1,000 samples. Also the colour coding of the t-SNE plot gives a slight indication of this difference in uncertainty. In contrast, for the ORGANISATION entity, there is a large uncertainty also for a training data set containing 1,000 samples. The bars that indicate mean uncertainty left in the data pool corroborate this difference.

The visualised differences in model uncertainty for different entities correspond to differences found in the evaluations against the gold standard, as shown in Figure 1. That is, the model trained to recognise ORGANISATION, which is visualised as uncertain, still yields a very low F-score when trained on 1,000 training samples. Similarly, that better results were achieved for PERSON and LOCATION when evaluating against the gold standard, is reflected by a visualisation that indicates a lower uncertainty for models trained on 1,000 training samples to detect these entity categories.

Conversely, the percentage of incorrect classifications increases when the training data set for the entity LOCATION increases. Thereby, the standard measurement, in the form of incorrect classifications when performing a cross-validation on the labelled data, fails to indicate changes in model performance.⁴

⁴This measure is equivalent to inverse accuracy. *Inverse* accuracy is used to match the uncertainty measure used, i.e.,

The spatial information in the t-SNE plot of word embeddings correspond well to differences with regards to the usefulness of embedding features between the three entity categories evaluated. That is, tokens classified as belonging to the categories PERSON and LOCATION, for which word embeddings were useful, are shown as clusters of red dots in the t-SNE plots. In contrast, tokens classified as belonging to ORGANISATION, for which word embeddings were shown not to be useful, mainly occur as scattered dots in the plot.

The output of experiments on the Japanese data, for a model trained on 138 manually labelled microblogs, is shown in Figure 3. Figure 3a visualises the state of the pool with regards to the LOCATION category, and Figure 3b shows pre-annotations resulting from this model.⁵

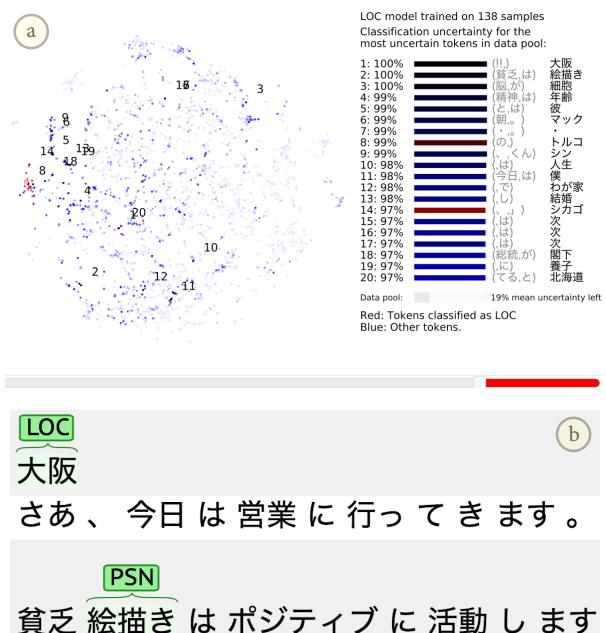


Figure 3: (a) The state for the LOCATION entity in the pool of unlabelled data, when the NER model has been trained on 138 manually labelled Japanese microblogs. Two potential entity clusters are shown in the t-SNE plot (close to 8, Turkey, and 20, Hokkaido). Which iteration is shown can be changed through the slider provided. (b) Pre-annotations for two samples selected for manual annotation, as they contain the two most uncertain tokens in the data pool, i.e., the tokens shown as the first two elements in the list of uncertain tokens.

the aim for both should be to reach 0%.

⁵The code for PAL, as well as for the experiments reported here, can be found at: <https://github.com/mariask2/PAL-A-tool-for-Pre-annotation-and-Active-Learning>. There, a link can also be found to a video showing how the state of the pool changes with an increasing training data size.

5 Discussion

Results for the LOCATION and PERSON entities yield that the combined functionality of active learning and incorporation of unsupervised features has the potential to lead to large increases in results on small data sets. This, in turn, shows that these techniques form useful components for the use case on which we focused here, i.e., to achieve models that can give acceptable performance on small data sets and that can be applied for providing pre-annotations when annotating larger data sets.

The categories LOCATION and PERSON seem to be relatively coherent in terms of the contexts in which they occur, as shown by the large model performance increases achieved when word embedding features were incorporated. In contrast, that slightly better results were achieved for ORGANISATION without word embedding features, indicates that entities belonging to this category occur in semantically diverse contexts.

These differences in context coherence between different entity categories were also shown by the t-SNE plot functionality, which we provided to meet one of the visualisation goals of the PAL tool extension of this study, i.e., the goal of showing whether the word embeddings used as features formed clusters corresponding to manually annotated entity categories. Thereby, the annotator is provided with a possibility to estimate the effect of these word embedding features in the active learning process.

The t-SNE plot and the bar charts of the extended version of the PAL tool also meet the visualisation goals of showing why a particular set of samples were chosen for annotation, and of showing how the increased size of the training data set affects the performance of the trained model. An increased training data size led to that two of the classifiers achieved an F-score that might be high enough to be acceptable for pre-annotation, while the F-score remained low for the ORGANISATION category, also when the data size was increased. These differences were reflected in the visualisations of the effects of the increased training data size.

We believe that visualisations that aim to increase the human understanding of the active learning process and of the features used, and that show how the state of the data pool changes as more data is manually annotated, have the potential to increase the intrinsic motivation for the annotation task. Future work will therefore include user

studies to determine how annotators perceive these visualisations that were added to the PAL tool, and how the visualisations affect the motivation for the annotation work. Such user studies should also include investigations of how the performance level of the machine learning model correlates with the perceived usefulness of the pre-annotations provided by the model.

6 Conclusion

We evaluated the ability of the PAL tool to reduce the training data size required through the use of active selection of data and through the incorporation of unsupervised features in the form of word embeddings. Results achieved for the categories LOCATION and PERSON showed that the combined functionality of active learning and incorporation of word embeddings has the potential to lead to large increases in results on small data sets. In contrast, word embeddings did not lead to any improvements in the performance for detecting the ORGANISATION entity, and low F-scores were achieved for this entity category, also when 1,000 samples were used for training the model.

The PAL tool was also extended with visualisation functionality, with the aim of increasing the human understanding of the active learning process and of the features used. The visualisations provided were able to indicate performance differences between the entities, as well as differences with regards to the usefulness of the embedding features. That is, the same differences that were shown in the formal evaluations against the gold standard annotations.

We hope that this study will inspire annotation projects to facilitate the annotation process by practically applying the methods that we have evaluated here. In particular, we hope that the application of PAL, and other tools that provide annotation support, will lead to that more annotation projects are being conducted on corpora for which crowdsourcing is not appropriate. For instance, corpora for specialised domains or smaller languages.

Acknowledgements

We would like to thank the reviewers for their valuable input. We would also like to thank the Swedish Research Council that funded this study (project numbers 2016-06681 and 2017-00626).

References

- [Baldwin et al.2015] Timothy Baldwin, Marie-Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 126–135, Beijing, China, July. Association for Computational Linguistics.
- [Copara et al.2016] Jenny Copara, Jose Eduardo Ochoa Luna, Camilo Thorne, and Goran Glava. 2016. Spanish NER with word representations and conditional random fields. In *Proceedings of the 6th Named Entities Workshop*, pages 34–40, Berlin, Germany. Association for Computational Linguistics.
- [Derczynski et al.2016] Leon Derczynski, Kalina Bontcheva, and Ian Roberts. 2016. Broad twitter corpus: A diverse named entity recognition resource. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1169–1179, December.
- [dos Santos and Guimarães2015] Cicero dos Santos and Victor Guimarães. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of the Fifth Named Entity Workshop*, pages 25–33, Beijing, China, July. Association for Computational Linguistics.
- [Dumitache et al.2013] Anca Dumitache, Lora Aroyo, Chris Welty, Robert-Jan Sips, and Anthony Levas. 2013. "Dr. Detective": combining gamification techniques and crowdsourcing to create a gold standard in medical text. In *Proceedings of the 1st International Workshop on Crowdsourcing the Semantic Web, Sydney, Australia, October 19, 2013*, pages 16–31, Aachen, Germany. CEUR-WS.org.
- [Godin et al.2015] Frédéric Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia lab @ acl w-nut ner sharedtask: named entity recognition for twitter microposts using distributed word representations. In *ACL 2015 Workshop on Noisy User-generated Text, Proceedings*, pages 146–153. Association for Computational Linguistics.
- [Guo et al.2014] Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 110–120.
- [Henriksson2015] Aron Henriksson. 2015. Learning multiple distributed prototypes of semantic categories for named entity recognition. *Int. J. Data Min. Bioinformatics*, 13(4):395–411, October.
- [Kucher et al.2017] Kostiantyn Kucher, Carita Paradis, Magnus Sahlgren, and Andreas Kerren. 2017. Active learning and visual analytics for stance classification with alva. *ACM Trans. Interact. Intell. Syst.*, 7(3):14:1–14:31, October.
- [Kudo2006] Taku Kudo. 2006. Mecab : Yet another part-of-speech and morphological analyzer. <http://mecab.sourceforge.jp>.
- [Lample et al.2016] Guillaume Lample, Miguel Baltes-teros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270.
- [Ling et al.2014] Wang Ling, Luis Marujo, Chris Dyer, Alan Black, and Isabel Trancoso. 2014. Crowd-sourcing high-quality parallel data extraction from twitter. In *Proceedings of the Ninth Workshop on Statistical Machine Translation, WMT '14, Stroudsburg, PA, USA*. Association for Computational Linguistics.
- [Lingren et al.2014] Todd Lingren, Louise Deleger, Katalin Molnar, Haijun Zhai, Jareen Meinzen-Derr, Megan Kaiser, Laura Stoutenborough, Qi Li, and Imre Solti. 2014. Evaluating the impact of pre-annotation on annotation speed and potential bias: natural language processing gold standard development for clinical named entity recognition in clinical trial announcements. *J Am Med Inform Assoc*, 21(3):406–13.
- [Martin et al.2017] David Martin, Sheelagh Carpendale, Neha Gupta, Tobias Hoßfeld, Babak Naderi, Judith Redi, Ernestasia Siahaan, and Ina Wechsung. 2017. Understanding the crowd: Ethical and practical matters in the academic use of crowdsourcing. In *Evaluation in the Crowd. Crowdsourcing and Human-Centered Experiments*, pages 27–69, Cham. Springer International Publishing.
- [Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.
- [Miller et al.2004] Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL HLT)*, pages 337–342, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Olsson2008] Fredrik Olsson. 2008. *Bootstrapping Named Entity Annotation by Means of Active Machine Learning*. Ph.D. thesis, University of Gothenburg. Faculty of Arts.

- [Pedregosa et al.2011] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- [Řehůřek and Sojka2010] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Paris, France, May. European Language Resources Association (ELRA).
- [Reimers and Gurevych2017] Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark, September. Association for Computational Linguistics.
- [Sahlgren2006] Magnus Sahlgren. 2006. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. thesis, Stockholm University.
- [Sang and Meulder2003] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147.
- [Schein and Ungar2007] Andrew I. Schein and Lyle H. Ungar. 2007. Active learning for logistic regression: an evaluation. *Mach. Learn.*, 68(3):235–265, October.
- [Settles2009] Burr Settles. 2009. Active learning literature survey. Computer Sciences Technical Report #1648, University of Wisconsin–Madison, <http://research.cs.wisc.edu/techreports/2009/TR1648.pdf>.
- [Shen et al.2004] Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics, ACL '04*, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Skeppstedt et al.2016] Maria Skeppstedt, Carita Paradiis, and Andreas Kerren. 2016. PAL, a tool for Pre-annotation and Active Learning. *JLCL*, 31(1):91–110.
- [Stenetorp et al.2012] Pontus Stenetorp, Sampo Pyysalo, Goran Topic, Tomoko Ohta, Sophia Ananiadou, and Jun’ichi Tsujii. 2012. BRAT: a web-based tool for NLP-assisted text annotation. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 102–107, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Tjong Kim Sang and Buchholz2000] Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132, Lisbon, Portugal.
- [Tomanek et al.2007] Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. Efficient annotation with the Jena ANnotation Environment (JANE). In *Proceedings of the Linguistic Annotation Workshop*, pages 9–16, Stroudsburg, PA, USA, June. Association for Computational Linguistics.
- [Turian et al.2010] Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July. Association for Computational Linguistics.
- [van der Maaten and Hinton2008] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data Using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.
- [Venhuizen et al.2013] Noortje Venhuizen, Valerio Basile, Kilian Evang, and Johan Bos. 2013. Gamification for word sense labeling. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)*, pages 397–403, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Yamada et al.2015] Ikuya Yamada, Hideaki Takeda, and Yoshiyasu Takefuji. 2015. Enhancing named entity recognition in twitter messages using entity linking. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 136–140, Beijing, China, July. Association for Computational Linguistics.
- [Yang et al.2016] Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *CoRR*, abs/1603.06270.

Creating Information-maximizing Natural Language Messages Through Image Captioning-Retrieval

Fabian Karl^{1,2} and Mikko Lauri¹ and Chris Biemann²

fabian.alexander.karl@gmail.com

lauri@informatik.uni-hamburg.de

biemann@informatik.uni-hamburg.de

¹Computer Vision, Department of Informatics, MIN, Universität Hamburg

²Language Technology, Department of Informatics, MIN, Universität Hamburg

Abstract

In this work, we propose the Image Captioning-Retrieval (ICR) problem that states the objective of language generation as information exchange. To solve the ICR problem, we design and implement an end-to-end neural network architecture that describes the content of images in natural language, and retrieves them solely based on these generated descriptions. The main goal is to be able to generate information-maximizing natural language messages. We experimentally show a strong increase in message information content while losing some grammatical correctness in the generated descriptions in a semi-supervised setting where caption generation is trained towards retrieval quality.

1 Introduction

Human thinking and reasoning are deeply connected to words and language. Turing (1950) famously defined the ability to hold a complex conversation as artificial intelligence. While this notion is debated (Searle, 1980), it is widely accepted that it is language that makes us human. An artificial system capable of producing human language will be received by us as human-like.

Current conversational and language producing systems can broadly be categorized into three classes: rule-based systems, supervised learning systems, and Reinforcement Learning (RL) models. Rule-based systems produce outputs by a set of conditionals and rules of varying complexity. This approach works well for expert systems and the understanding of simple commands. Due to

the predictability and traceability, rule-based language systems dominate commercial applications. Supervised learning systems apply supervised optimization strategies to predict appropriate language outputs for given inputs (Vinyals and Le, 2015). A prerequisite is a corpus of conversational training examples containing input sentences and corresponding output sentences. RL-based conversational systems (English and Heeman, 2005; Li et al., 2016) seek to learn a dialog policy that guides how the artificial agent should follow when interacting with a user.

While current state-of-the-art systems are arguably able to produce language that seems human-like, their objective is stated as mere production of well-sounding sentences. However, production of grammatically correct sentences as an end goal falls short of the motivation humans have for language production, namely the exchange of information (Kirby, 2007). In Mathur and Singh (2018) it is noted that especially sequence-to-sequence models cannot solve the language modelling problem, since "*the objective function that is being optimized does not capture the actual objective achieved through human communication, which is typically longer term and based on exchange of information rather than next step prediction*". The main driver of a conversational system should not be the direct production of sentences in a human-readable language, but the optimal amount of information exchange between agents (Steels, 2015).

In this work, we examine language generation through an alternative objective of maximum information exchange. We propose to train a language production system directly with the motivation of maximizing information content, rather than using language modelling objectives. To achieve this,

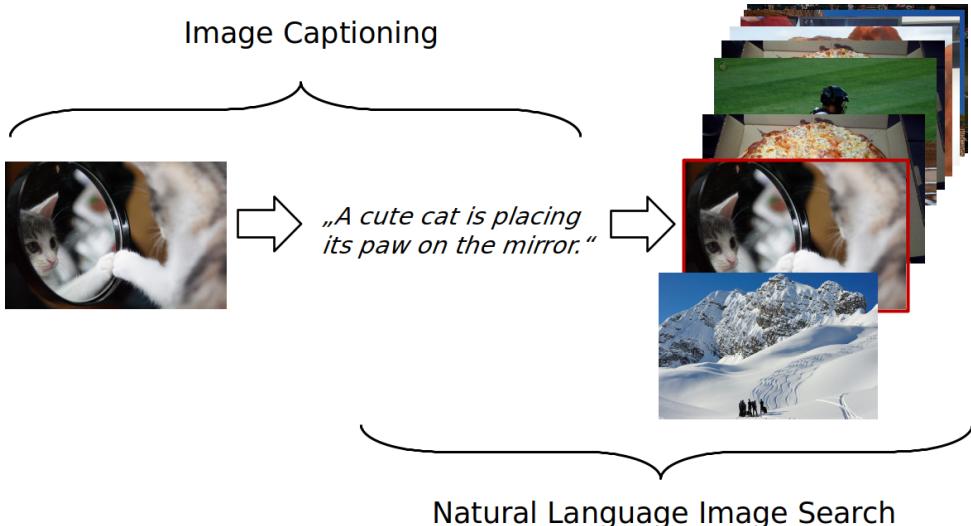


Figure 1: The Image Captioning-Retrieval (ICR) problem simulates a natural language message passed from one agent to another, and is composed of Image Captioning (IC) and Natural Language Image Search (NLIS).

we propose the Image Captioning-Retrieval (ICR) problem. The ICR problem simulates a message passed from one agent to another, and is composed of two parts: Image captioning (IC) and natural language image search (NLIS) as illustrated in Figure 1. IC describes or captions a given image with a sentence in natural language. NLIS takes the caption as an input and retrieves the closest image out of a set of candidate images. By combining IC and NLIS, we can train our language production system directly with the motivation of information exchange. The constraint that the communication takes place in human-understandable language is ensured by producing captions in natural language. For this, we first pre-train the IC system in a supervised fashion using pairs of images and captions, and subsequently continue to train the overall system on the retrieval task. This can be viewed as a semi-supervised setting since captions are improved not through direct supervision on gold captions, but on indirect supervision on discriminating between pictures in the retrieval task.

Our contribution is two-fold. Firstly, we show that solving the ICR problem gives rise to natural language messages, while experimentally showing a strong increase in message information content. Secondly, we qualitatively present that the descriptions generated by our model capture more details of images as compared to plain IC systems.

The remainder of the paper is organized as follows. In Section 2, we review relevant related works in image captioning, natural language image

search and neural learning architectures. Section 3 describes our overall approach, detailing the respective subsystems and their combination. The experimental setup is laid out in Section 4, before reporting quantitative evaluation results in Section 5. Qualitative observations are discussed in Section 6, Section 7 draws conclusions and provides directions for further work in natural language learning through conversations.

2 Related Work

State-of-the-art natural language production systems apply supervised learning, in particular the sequence-to-sequence model of Vinyals and Le (2015). This approach was inspired by machine translation (Sutskever et al., 2014), and has since been replicated multiple times. While an in-depth survey of natural language generating systems is beyond the scope of the present paper, we direct the interested reader to the recent survey of Gatt and Krahmer (2018). In our subsequent review, we discuss the two key subtasks of our ICR problem (Fig. 1), IC and NLIS, and the interplay of systems solving these two tasks.

Given an input image, an IC system outputs a description of the image in natural language. In turn, given as input a textual description of an image, an NLIS system finds the image that best matches the input description among a set of candidate images. We review techniques and ideas most closely related to our focus on the information exchange mo-

tivation for language generation. These approaches typically combine an IC network and an NLIS network and train them jointly. For a recent general survey of deep learning techniques applied to IC, we refer the reader to Hossain et al. (2019).

Most related to our work, the idea of scoring image descriptions based on the amount of information carried in the sentence is proposed in Hodosh et al. (2013). Instead of using traditional n-gram based evaluation measures like the BLEU (Papineni et al., 2002) or the CIDEr score (Vedantam et al., 2015), Hodosh et al. (2013) propose to use an NLIS system, pre-trained on human-annotated image-caption-pairs, to score the created image captions. The idea is widely used in other recent works in IC (Devlin et al., 2015; Vinyals et al., 2017; Karpathy and Fei-Fei, 2017; Donahue et al., 2017). The general architecture of these models contains an IC encoder-decoder model that encodes image information into textual form, and an image scoring system that evaluates the created captions using an NLIS system. The IC model is often a combination of a convolutional neural network (CNN) and a long-short-term memory network (LSTM).

Adversarial training is employed by several state-of-the-art works in IC (Dai et al., 2017; Liang et al., 2017; Liu et al., 2018). An NLIS network is applied to discriminate between generated and real samples. In Shetty et al. (2017), the objective is altered from merely reproducing ground truth captions to matching a distribution of human generated captions by applying an approximate Gumbel sampler.

RL is employed in some recent approaches such as the method by Ren et al. (2017b). A reward function is derived by considering visual-semantic embedding similarities: input images and captions both are mapped into a embedding space, and their similarity in this space is measured by an appropriate metric.

In contrast to the reviewed work we explicitly define information exchange as the primary objective for IC and NLIS. Through this we clearly separate us from related studies that use information exchange merely as a performance indicator or a general guidance.

3 Image Captioning Retrieval Network

Our ICR network is an IC network and an NLIS network, combined by a Gumbel softmax layer.

3.1 Image Captioning

The IC model receives an image and returns multiple probability distributions over a vocabulary.

The input for the model is an image $x^{im} \in \mathbb{R}^{h \times w \times c}$, where h, w, c are the height, width and color dimension, together with a sequence of words. The model output is a probability distribution over a fixed vocabulary V . Each word is thus assigned a likelihood of being the next word.

The input image is resized to a fixed size and fed through an image encoder (e.g. CNN) with the parameters θ_ϕ that extract the most important image features in a vector $\phi(x^{im}, \theta_\phi) \in \mathbb{R}^k$, where k is the length of the feature vector.

The respective image annotation is embedded in a dense word embedding, yielding the second model input $x^{se} \in \mathbb{R}^{t \times d}$, where t is the number of words in a sentence and d is the dimensionality of the dense word embedding. The embedded sentence is fed through a sentence encoder (e.g. LSTM) resulting in a $t \times l$ tensor, where l is the length of the feature vector.

Now x^{im} is replicated t times and concatenated with the sentence features. This results in a $t \times (l + k)$ tensor, which is fed through a block of fully connected layers and a final softmax layer, squeezing the model output into t probability distributions with $P(y_t | x_{1 \rightarrow t-1}^{se}, \phi(x^{im}, \theta_\phi))$, where y_t is the probability over the vocabulary V at timestep t , x^{se} is the information from the previous words and $\phi(x^{im}, \theta_\phi)$ is the image vector.

At training time, x^{se} and the target $y \in \mathbb{R}^{t \times d}$, with the same shape as x^{se} , are representations of the same *ground truth* sentence. This training technique is called teacher forcing. x^{se} is shifted one time-step into the future by adding a *start-symbol* at its beginning. This way, word y_t equals x_{t+1}^{se} and the model is trained to predict the next word of the same sentence x^{se} . An *end-symbol* is appended to y , so input and output have the same length and the model is trained on how to end the sentence. The loss is calculated through the cross-entropy of the predicted probability distribution and the *ground truth* distribution. This allows a quick and stable learning process but also leads to the so-called *exposure bias* (Ranzato et al., 2016).

At inference, only the image vector $\phi(x^{im}, \theta_\phi)$ is available. The model starts with \hat{x}^{se} , containing only the *start-symbol*, as first input and generates $P(\hat{y}_t)$. Depending on the selection mode, one word y_t from $P(\hat{y}_t)$ is selected and appended to the pre-

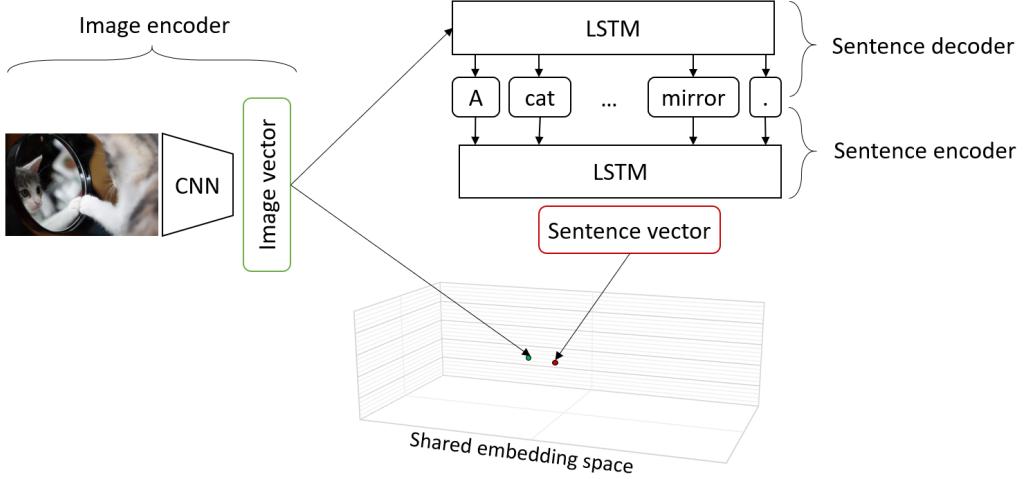


Figure 2: Our ICR model. Annotations and images are both transformed into feature representations, which are mapped into a shared embedding space. The distance in this space defines the similarity of the image-annotation pair.

vious input $\hat{x}_{1 \rightarrow t-1}^{se}$. It then serves as new input for the next prediction step.

3.2 Natural Language Image Search

Our NLIS model is realized through an image and a sentence encoder that are trained on the triplet ranking loss (Karpathy et al., 2014; Ren et al., 2017a; Karpathy and Fei-Fei, 2017; Wang et al., 2017; Faghri et al., 2018; Liu et al., 2018).

Both encoders are similar to the ones used in our IC model. Images x^{im} are transformed into feature representations $\phi(x^{im}, \theta_\phi) \in \mathbb{R}^{d_\phi}$, where ϕ is the image encoder (e.g. CNN), with model parameters θ_ϕ . Correspondingly, sentences x^{se} are embedded and transformed into a feature representation through a model $\psi(x^{se}, \theta_\psi) \in \mathbb{R}^{d_\psi}$, where ψ is the sentence encoder (e.g. LSTM) with model parameters θ_ψ .

$$f_{im}(x^{im}, W_{im}, \theta_\phi) = \|W_{im}^T \phi(x^{im}, \theta_\phi)\|_2 \quad (1)$$

$$f_{se}(x^{se}, W_{se}, \theta_\psi) = \|W_{se}^T \psi(x^{se}, \theta_\psi)\|_2 \quad (2)$$

Both feature representations are mapped into a shared embedding space of size e by linear projection with weight matrices $W_{im} \in \mathbb{R}^{d_\phi \times e}$ and $W_{se} \in \mathbb{R}^{d_\psi \times e}$. The resulting projections are normalized with the L2 norm to lie on the unit hypersphere.

$$s(im, se) = f_{im}(x^{im}, W_{im}, \theta_\phi) \cdot f_{se}(x^{se}, W_{se}, \theta_\psi) \quad (3)$$

The similarity between an image-sentence pair is defined as the inner product between the two

normalized vectors, resulting in the cosine similarity (Subhashini and Kumar, 2010).

$$\mathcal{L}(\theta, B_{im}, B_{se}) = \frac{1}{N} \sum_{n=1}^N L(im_n, se_n, B_{im'}, B_{se'}) \quad (4)$$

For a batch of images, $B_{im} = \{x^{im}\}_{n=1}^N$, and corresponding sentences, $B_{se} = \{x^{se}\}_{n=1}^N$, the batch loss is calculated by comparing every image against every sentence and vice versa. In every iteration, one image-sentence pair is selected as *true* pair, marked as (im_n, se_n) . The similarity of this pair is compared to the similarities between the image and all other sentences or the sentence and all other images respectively. A batch of sentences, without the correct sentence, is denoted as $B_{se'}$ and a batch of images without the correct image as $B_{im'}$.

All possible parameters to be optimized are defined by $\theta = \{\theta_\phi, \theta_\psi, W_{im}, W_{se}\}$. Depending on the experimental setup, however, θ_ϕ and/or W_{im} are not optimized or finetuned.

$$\begin{aligned} L_{SH}(im, se, \hat{im}, \hat{se}) = & \\ & \sum_{\hat{se}} [\alpha - s(im, se) + s(im, \hat{se})]_+ + \\ & \sum_{\hat{im}} [\alpha - s(im, se) + s(\hat{im}, se)]_+ \end{aligned} \quad (5)$$

L_{SH} is defined as the sum of hinges and describes the *classic* triplet ranking loss. Let α be the margin that the similarity of all wrong image-sentence pairs should be smaller than the similarity of the

correct image-sentence pair. $s(im, se)$ describes the similarity of the correct image-sentence pair while $s(im, \hat{se})$ describes the similarity between an incorrect image-sentence pair. In order to avoid negative losses, we use positive values only, as defined by the notation $[x]_+ = \max(0, x)$. The second term is symmetrical to the first term. In the first term, an image is fixed and the similarity with different candidate sentences is calculated and returned. In the second term, a sentence is fixed and all other images are iterated over to calculate the similarities. Faghri et al. (2018) report a steep increase in accuracy on the NLIS task when using triplet ranking loss with the max of hinges, L_{MH} . This refers to selecting the one (negative) sample with the highest loss in every mini-batch. The only difference between L_{MH} and L_{SH} is the selection of the biggest error, $\max_{\hat{se}} [\alpha - s(im, se) + s(im, \hat{se})]_+$, instead of the summation of errors, $\sum_{\hat{se}} [\alpha - s(im, se) + s(im, \hat{se})]_+$.

3.3 Image Captioning-Retrieval

Our ICR network is a combination of the two models described above. In order to overcome the problem of discrete word representations being not differentiable, the Gumbel softmax trick (Jang et al., 2016) is used to transform one-hot probability distributions into pseudo-one-hot-representations.

The original Gumbel-Max trick (Gumbel, 1954) is a simple and efficient way to draw samples from a categorical distribution with class probabilities π . $g \in (0, 1)$ is called the Gumbel distribution and is calculated from u , drawn from a uniform distribution between 0 and 1.

$$g = -\log(-\log(\text{Uniform}(0, 1))) \quad (6)$$

$$z = \text{one hot} \left(\underset{i}{\text{argmax}} [g_i + \log(\pi)] \right) \quad (7)$$

Since argmax is non-differentiable, the continuous softmax function is used as an approximation. τ is the temperature of the softmax. The smaller τ is, the closer the distribution is to a one-hot encoding. y_i is the resulting k-dimensional word distribution.

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^k \exp((\log(\pi_j) + g_j)/\tau)} \quad \text{for } i, \dots, k \quad (8)$$

A second challenge is the sampling of novel sentences. Our ICR model needs a complete input

sentence x^{se} to be able to determine the probability for every sub-sentence $x_{t_1:t_i}^{se}$. This can either be achieved by creating complete and novel sentences with our IC model in a pre-step or by directly using the Gumbel softmax trick in this phase. Since the Gumbel softmax activation function introduces randomness into the selection process, unseen word combinations can occur, from which the model will not be able to recover. For this reason we decided to use the first-mentioned approach.

When feeding the novel image annotation through our ICR model, it will be fed through the IC model again and reproduce the output \hat{y} . The output is transformed with the Gumbel softmax activation function, which selects one word randomly based on its probability and transforms it into a value close to one. All other words will receive a very low probability, close to 0. Let $\gamma(\hat{y})$ be the Gumbel softmax output.

Together the original image vector $\phi(x^{im}, \theta_\phi)$ and $\gamma(\hat{y})$ are fed into the NLIS network to output a similarity matrix, containing similarities between every image and every sentence. From this similarity matrix, either the sum or the max of hinges loss (Section 3.2) can be calculated and used for training.

4 Experimental Setup

Our experiments are designed to optimize information exchange between the IC and the NLIS system. Information exchange is measured by the image retrieval score, which is reported in the percentage of images ranked within the best 1, 5 or 10 ranked images ($r@1$, $r@5$, $r@10$). The Consensus-based Image Description Evaluation (CIDEr) (Vedantam et al., 2015) score for the generated annotations is presented alongside. CIDEr is an n-gram based evaluation metric especially created for image annotation.

We use MSCOCO dataset with 2017 split (Lin et al., 2014; Chen et al., 2015) for training and validation. The dataset contains 118,287 training and 5,000 validation images, all of them annotated with five ground truth sentences.

In preprocessing, all annotations are cut or padded to contain exactly 16 tokens. Tokens appearing less than 10 times are replaced with the *unknown word* token. Every word is embedded with a pre-trained English fasttext model (Bojanowski et al., 2017). All images were encoded by extracting the last fully connected layer of ResNet50

Table 1: Performance of our IC and NLIS model after stand-alone pre-training on their respective task (*Our), compared to VSA (Karpathy and Fei-Fei, 2017), UVS (Kiros et al., 2014), VSE++ (Faghri et al., 2018), sm-LSTM (Huang et al., 2017), m-RNN (Mao et al., 2014) and LRCN (Donahue et al., 2017) on different measures as reported in the literature. NLIS results refer to 1,000 test images and 5,000 respective descriptions from MSCOCO 2017. r@n shows the percentage of sentences/images ranked under the top n ranks. BLEU4 and CIDEr are received from the C40 test set of the official 2015 COCO Caption Challenge Competition. Results with most similar architectures are listed if available.

System	Image captioning			Image retrieval		
	r@1	r@5	r@10	r@1	r@5	r@10
VSA	38.4	69.9	80.5	27.4	60.2	74.8
*Our	39.9	69.8	80.1	32.0	66.3	80.8
UVS	43.4	75.7	85.8	33.0	67.2	80.6
VSE++	43.6	74.8	84.6	33.7	68.8	82.0
sm-LSTM	53.2	83.1	91.5	40.7	75.8	87.4

System	Image captioning		
	BLEU4	CIDEr	
VSA	0.446	0.692	
*Our	0.472	0.753	
UVS	0.517	0.752	
m-RNN	0.578	0.896	
LRCN	0.585	0.934	

(2,048 nodes), pre-trained on ImageNet (Deng et al., 2009).

IC and NLIS network are separately pre-trained until they yield optimal annotation and ranking results. Multiple hyperparameters (model architecture, number of epochs, learning rate, etc.) of both models were empirically optimized to yield results close to state-of-the-art performance for their respective task. For both models, sentences are encoded by 1,024 LSTM cells. Images are projected onto vectors of the same size with a dense layer. In our NLIS network, encoded sentence features are also projected onto a 1,024 dimensional space by a fully connected layer. In our IC model, sentence and image features are concatenated and fed through two dense layers (1,024 and 2,048 nodes), before the final softmax layer. Between every layer we added dropout layers with 0.4 dropout to prevent the model from overfitting. When training our NLIS model we used *sum* of hinges for one epoch before switching to *max* of hinges loss. This was necessary for a stable training. In all later ICR experiments we used *max* of hinges loss.

Table 1 shows the performance of our models

after pre-training compared to related studies that used similar techniques with similar network architectures. The performance of our NLIS model builds the baseline for further training with our complete ICR model. In order to combine IC and NLIS model in our final model, we implemented both models in the same framework. Simply reusing models from related work was not possible due to the incompatibility of different neural network frameworks.

In our main training loop, 20,000 images are randomly selected per epoch and fed through our ICR network. A loss is calculated for the generated annotation and for the retrieved image. Mini-batch size is set to 128 for all experiments. The model was trained for 40 epochs with Adam as optimizer. The learning rate is set to 0.0002 for the first 20 epochs and then decreased to 0.00002 for the rest of the training process.

Optimizing all weights in the ICR network leads to an unstable training process and often resulted in sudden drops in performance. Freezing the weights of the image projection layer from the beginning of training (IP=F) or at a certain epoch (IP=17) stabilized the training process. Freezing the weights of the sentence encoder (SE=F) had a similar stabilizing effect on the training. Training with only self-generated sentences right from the start leads to an instant decrease in performance since the model has no time to adjust to flawed input sentences. To counter this issue, novel self-generated annotations are slowly added to existing ground truth sentences. This is implemented by randomly selecting an annotation from a list of both ground truth annotations and generated ones. In the beginning, this list contains only ground truth samples. At every epoch, novel annotations are added. When the list reaches a defined size (INF=5, 10, 15), a random sentence is dropped from the list. This way, novel sentences are slowly infused into the training process.

To increase ranking performance, true image-sentence pairs were added to the output from the IC network. In this case, one mini-batch contains 64 image-sentence pairs generated by our IC network and 64 true image-sentence pairs directly from the dataset (TP=T). Otherwise the whole mini-batch contains only self-generated samples (TP=F). Both methods result in a 128×128 similarity matrix for one mini-batch. After the training phase, 1,000 validation images are captioned and retrieved to

Table 2: Ranking retrieval results for different experimental settings on 1000 validation images from MSCOCO 2017. *TP=True Pairs, SE=Sentence encoders trainable, IP=Image Projection layer trainable or trained until which epoch, INF=Infusion list size, NLIS sum=Sum over all image scores, C=CIDEr*

TP	SE	IP	INF	Sentence Retrieval			Image retrieval			NLIS sum	C
				r@1	r@5	r@10	r@1	r@5	r@10		
F	T	F	10	34.7	72.1	86.9	33.2	69.7	83.7	186.6	0.061
F	F	F	10	40.8	77.8	89.9	38.8	76.3	88.9	204.0	0.101
T	T	17	10	44.4	79.9	90.5	40.8	79.1	89.7	209.6	0.049
T	F	T	10	47.6	84.2	93.6	43.1	81.8	92.5	217.4	0.094
T	F	T	15	46.2	86.4	93.6	46.0	83.0	93.0	222.0	0.083
Pre-training Baseline				39.9	69.8	80.1	32.0	66.3	80.8	179.1	0.753

determine the performance of our model.

5 Results

Table 2 shows various experimental settings and their resulting ranking and CIDEr score. In the last row, the baseline ranking and annotation performance is reported. It represents our best performance of the two models when trained on their respective tasks alone.

The table shows that the usage of true image pairs (TP) generally increases the ranking performance of the network. The best experimental results were observed when freezing the sentence encoder weights for the ICR training (SE=F) but not the image projection layer (IP=T). An infusion list size of 10 (INF=10) yields optimal sentence retrieval scores while an infusion list size of 15 (INF=15) results in a 3 percent-point increase in the r@1 for the NLIS score and the best overall retrieval score (NLIS sum). Training runs with no infusion list (not mentioned in Table 2) were abandoned early in the experimental phase, for they resulted in unstable training and worse ranking scores than our baseline.

Compared to the ranking performance of our baseline (Table 1), we observe improvements for all reported experiments. Under the same evaluation set (1000 validation images), our best model improves image r@1 results by 14.0 percentage points resulting in 46.0% correctly retrieved images through our self-generated image descriptions. 80.0% of all described images were retrieved within the 10 top ranks. Not only could we increase our retrieval performance immensely compared to our baseline, but we also outperform all related studies using similar image encoders. This indicates that our self-generated sentences contain more image information than the *ground-truth* an-

notations, created by human annotators. CIDEr scores, however, decrease from our baseline performance of 0.72 to around 0.10.

The increase in retrieval scores and the decrease in CIDEr can be observed in Figure 3 as well. It shows a selection of images and different annotations. The first annotation is the annotation generated by our IC system, after pre-training (PT). GT shows one of the *ground truth* captions for comparison. The last sentence is the generated description from our best performing (ICR) model. Word repetitions, missing of stop-words and the selection of more specific and precise words (e.g. locomotive instead of train) are at the same time responsible for higher retrieval scores and lower CIDEr score. Since n-gram based evaluation metrics use direct comparison between prediction and ground-truth sentence, using often occurring words (e.g. stop-words) and general terminology (e.g. train) normally yields better results. Ironically, these words often carry the least amount of information.

6 Discussion

A comparison between the images in Figure 3 and their descriptions after the pre-training phase and after the ICR training phase shows that the increase in information exchange is not only visible in the ranking scores, but also leads to arguably better generated descriptions.

The sentences created after the pre-training are almost exclusively grammatically correct and describe the image content more or less accurately. Generated descriptions show less grammatical structure after the IC system was trained to maximize the ranking performance, but the content of the sentence describes the image in much more detail and correctness.

The generated sentences after ICR training often



PT: a train is parked on the tracks in a city .

GT: A train traveling past tall white buildings on train tracks.

ICR: an old locomotive train caboose on an railroad train tracks wires wires wires wires overpass overpass



PT: a man is holding a frisbee in his hand .

GT: A woman wearing glasses holding a tennis racket.

ICR: this man holding his tennis racket wearing his neck shirt shirt shirt sleeve sleeve wrist wrist



PT: a man is playing tennis on a court .

GT: Two men shaking hands while standing on a tennis court.

ICR: two men man standing on an tennis court on an tennis court net courts courts courts



PT: a man riding on the back of an elephant .

GT: Man riding an elephant up a hill near a field.

ICR: two man riding on elephant elephant walking through some river river saddle a river stream river

Figure 3: Next to every image, the description generated by the pre-trained IC model (PT), one of the ground truth descriptions (GT) and the descriptions, generated after training the ICR model.

contain repeating words, and they do not contain the *end-symbol* anymore. Both of these effects are likely due to the pre-training of the system. During the pre-training phase, only correct sentences were used as input for the model. In the ICR training phase, new sentences are generated and used for training. Additionally, since the Gumbel softmax trick is a statistical sampling method, the word with the highest probability is not always picked, as it has been before with greedy picking. This means the system encounters new situations that it has to deal with. Since it was trained with teacher-forcing, it has developed little robustness against these novel situations. Interestingly, the ICR system tries to fully use the maximum length of 16 tokens, possibly conveying the importance of image elements with word repetition.

It is important to mention that the grounding between words and entities in the images stays intact during the training. This means, the network keeps using the same words for certain scenes or objects, learned in the pre-training phase. This is highly relevant for a system trying to learn language without explicit targets. It means that the system keeps connections between image entities and words, even when trained on a different task. This allows us to focus on a more implicit goal like information exchange.

Regarding the first image in Figure 3, one can see, that the description after the ICR training includes "*an old locomotive*" instead of only "*a train*". The description also contains "*wires overpass*", describing the electrical wires over the train, even though this information was not present in any of the 5 human annotated sentences. This shows that the model is no longer explicitly trained on the *true* sentences, but has a much more implicit objective. In order to optimize the ranking performance, additionally, highly distinct image information is reflected in the wording. The fourth image in Figure 3 shows similar increases in content and detail description. The information that the elephant is "*walking through some river*" is crucial to distinctly rank this image higher than other elephant images.

In the third picture, the new description is less general. The pre-trained system is producing a generic sentence, more or less fitting to any tennis scene. The description, generated after the ICR training is more accurate in its context. The same is true for images 2 and 3. In general, the image content is described in more detail and in more accuracy. The sentences are less grammatical than before, however.

These findings are satisfying and show that our objective trains our system to transfer information while still creating human-readable sentences. The fact that the created sentences are still grounded show that our language system, once pre-trained, keeps its relations between objects and words intact. Our main goal of increasing the amount of exchanged information is clearly reached. Our secondary goal of insuring the human-readability of the generated language is partly satisfied and could be addressed with future work.

7 Conclusion

We clearly show how training an IC network with a more implicit objective like the ranking results

from our NLIS network can improve the amount of information captured in the generated sentences. The newly generated sentences are not grammatically perfect but understandable by humans. More importantly, after training our ICR model, generated descriptions capture more distinct details of images and describe more aspects of the images. The ranking performance was increased by a large margin, surpassing previous image search approaches.

This work has strengthened our belief that language generation and comprehension learning can benefit from implicit objectives in a joint learning setup as opposed to learn them from explicit supervision separately. Language offers a mapping from a high dimensional to discrete space. It offers the exchange of complex information in an equally complex but agreed-upon system. If information exchange is a major goal, more effort should be placed in implicitly modeling, with objectives like information exchange in order to solve tasks, requiring content that can only be transferred by language. The proposed language game in this work builds one of the most basic language games: describing and finding an image.

More sophisticated games, like solving riddles, answering questions, walking through a maze or executing commands can all be implemented based on language instructions. These games all have to be designed in a way that succeeding is a direct implication of information exchange. If this approach is used, while language grounding and correct grammar are enforced and guaranteed for, we will have a chance of optimizing language generation and comprehension directly on target tasks, which should result in more targeted and better-suited systems as opposed to training on auxiliary objectives.

In future work, conversation generation can also be targeted. The challenge there is that conversation should only be as informative as required in a given situation to not distract or cause an unnecessarily high cognitive load.

References

- P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.
- X. Chen, H. Fang, T.Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. 2015. Microsoft COCO Captions: Data Collection and Evaluation Server. *Computing Research Repository*, abs/1504.00325.
- B. Dai, S. Fidler, R. Urtasun, and D. Lin. 2017. Towards Diverse and Natural Image Descriptions via a Conditional GAN. In *International Conference on Computer Vision*, pages 2989–2998, Venice, Italy.
- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, Miami Beach, MI, USA.
- J. Devlin, H. Cheng, H. Fang, S. Gupta, L. Deng, X. He, G. Zweig, and M. Mitchell. 2015. Language Models for Image Captioning: The Quirks and What Works. In *Conference on Empirical Methods in Natural Language Processing*, pages 100–105, Lisbon, Portugal.
- J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell. 2017. Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691.
- M. English and P. A. Heeman. 2005. Learning Mixed Initiative Dialog Strategies By Using Reinforcement Learning On Both Conversants. In *Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 1011–1018, Vancouver, BC, Canada.
- F. Faghri, D. J. Fleet, R. Kiros, and S. Fidler. 2018. VSE++: Improved Visual-Semantic Embeddings. In *Proceeding of the British Machine Vision Conference*, Newcastle upon Tyne, UK.
- A. Gatt and E. Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.
- E. J. Gumbel. 1954. *Statistical theory of extreme values and some practical applications; a series of lectures*. U.S. Government Printing Office, Washington, D.C., USA.
- M. Hodosh, P. Young, and J. Hockenmaier. 2013. Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. *Journal of Artificial Intelligence Research*, 47:853–899.
- MD. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga. 2019. A comprehensive survey of deep learning for image captioning. *ACM Comput. Surv.*, 51(6):118:1–118:36.
- Y. Huang, W. Wang, and L. Wang. 2017. Instance-Aware Image and Sentence Matching with Selective Multimodal LSTM. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7254–7262, Honolulu, HI, USA.

- E. Jang, S. Gu, and B. Poole. 2016. Categorical Reparameterization by Gumbel-Softmax. *Computing Research Repository*, abs/1611.01144.
- A. Karpathy and L. Fei-Fei. 2017. Deep Visual-Semantic Alignments for Generating Image Descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):664–676.
- A. Karpathy, A. Joulin, and L. F. Fei-Fei. 2014. Deep Fragment Embeddings for Bidirectional Image Sentence Mapping. In *Advances in Neural Information Processing Systems*, pages 1889–1897, Montréal, QC, Canada.
- S. Kirby. 2007. The evolution of language. *Oxford Handbook of Evolutionary Psychology*, pages 669–681.
- R. Kiros, R. Salakhutdinov, and R. S. Zemel. 2014. Unifying Visual-Semantic Embeddings with Multi-modal Neural Language Models. *Computing Research Repository*, abs/1411.2539.
- J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao. 2016. Deep Reinforcement Learning for Dialogue Generation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, TX, USA.
- X. Liang, Z. Hu, H. Zhang, C. Gan, and E. P. Xing. 2017. Recurrent Topic-Transition GAN for Visual Paragraph Generation. In *International Conference on Computer Vision*, pages 3382–3391, Venice, Italy.
- T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. 2014. Microsoft COCO: Common Objects in Context. *Computing Research Repository*, abs/1405.0312.
- X. Liu, H. Li, J. Shao, D. Chen, and X. Wang. 2018. Show, tell and discriminate: Image captioning by self-retrieval with partially labeled data. In *Proceedings of the 15th European Conference on Computer Vision - ECCV 2018*, pages 353–369, Munich, Germany.
- J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille. 2014. Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN). *Computing Research Repository*, abs/1412.6632.
- V. Mathur and A. Singh. 2018. The rapidly changing landscape of conversational agents. *Computing Research Repository*, abs/1803.08419.
- K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, USA.
- M. A. Ranzato, S. Chopra, M. Auli, and W. Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, San Juan, Puerto Rico.
- Z. Ren, H. Jin, Z. Lin, C. Fang, and A. Yuille. 2017a. Multiple instance visual-semantic embedding. In *Proceedings of the British Machine Vision Conference*, London, UK.
- Z. Ren, X. Wang, N. Zhang, X. Lv, and L.-J. Li. 2017b. Deep Reinforcement Learning-Based Image Captioning with Embedding Reward. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1151–1159, Honolulu, HI, USA.
- J. R. Searle. 1980. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3):417424.
- R. Shetty, M. Rohrbach, L. A. Hendricks, M. F., and B. Schiele. 2017. Speaking the Same Language: Matching Machine to Human Captions by Adversarial Training. In *International Conference on Computer Vision*, pages 4155–4164, Venice, Italy.
- L. Steels. 2015. *The Talking Heads experiment: Origins of words and meanings*. Language Science Press, Berlin, Germany.
- R. Subhashini and V. J. S. Kumar. 2010. Evaluating the performance of similarity measures used in document clustering and information retrieval. In *International Conference on Integrated Intelligent Computing*, pages 27–31, Bangalore, India.
- I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, Montréal, QC, Canada.
- A. M. Turing. 1950. Computing machinery and intelligence. *Mind*, 49:433–460.
- R. Vedantam, C. L. Zitnick, and D. Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575, Boston, MA, USA.
- O. Vinyals and Q. V. Le. 2015. A Neural Conversational Model. *Computing Research Repository*, abs/1506.05869.
- O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. 2017. Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):652–663.
- L. Wang, Y. Li, and S. Lazebnik. 2017. Learning Two-Branch Neural Networks for Image-Text Matching Tasks. *Computing Research Repository*, abs/1704.03470.

German End-to-end Speech Recognition based on DeepSpeech

Aashish Agarwal and Torsten Zesch
Language Technology Lab
University of Duisburg-Essen
Duisburg, Germany

Abstract

While automatic speech recognition is an important task, freely available models are rare, especially for languages other than English. In this paper, we describe the process of training German models based on the Mozilla DeepSpeech architecture using publicly available data. We compare the resulting models with other available speech recognition services for German and find that we obtain comparable results. Acceptable performance under noisy conditions would, however, still require much more training data. We release our trained German models and also the training configurations.

1 Introduction

Automatic speech recognition (ASR) is the task of translating a spoken utterance into a textual transcript. It is a key component of voice assistants like Google Home (Li et al., 2017), in spoken language translation devices (Krstovski et al., 2008), or for automatic transcription of audio and video files (Liao et al., 2013). For any language beyond English, readily available pre-trained models are still rare. For German, we are only aware of the model by Milde and Köhn (2018) for the Kaldi framework (Povey et al., 2011). For the recently introduced Mozilla DeepSpeech framework, a German model is still missing. This is a serious obstacle to applied research on German speech data, as available web-services by Google, Amazon, or Microsoft are problematic due to data privacy reasons. We thus use publicly available speech data to train a German DeepSpeech model. We release our trained German model and also publish the code and configurations enabling researchers to (i) directly use the model in applications, (ii) reproduce state-of-the-art results, and (iii) train new models based on other source corpora.

2 Speech Recognition Systems

Due to the underlying complexity of recognizing spoken language and the wish of the service provider to keep the model private, many systems are offered as *web services*. This includes commercial services like Google Cloud Speech-to-Text (He et al., 2018), Amazon Alexa Voice Services¹, IBM Watson Speech to Text (Saon et al., 2017) or Speechmatics² as well as academic services like BAS.³ While web services are convenient, there are many situations where they cannot be used:

- sending data to a web service might violate data privacy protection laws
- as the data throughput of a web service is limited; it might rule out batch processing of large amounts of speech data
- the user cannot control (or change) the functionality of a remotely deployed web service
- research results based on web service calls are not easily replicable, as services might change without notice or become unavailable altogether.

For this work, we therefore consider only frameworks that can be used locally and without restrictions. One such framework is **Kaldi** (Povey et al., 2011) which was found to be the best performing open-source ASR system in a previous study (Gaida et al., 2014). It is open-source toolkit written in C++ that supports conventional models (e.g. Gaussian Mixture Models) as well as deep neural networks. Recently, end-to-end neural systems like **wav2letter++** (Pratap et al., 2018) provided by Facebook, or **DeepSpeech**⁴ provided by Mozilla have been introduced. To our knowledge, there is

¹<https://developer.amazon.com/alexa/science>

²<https://www.speechmatics.com>

³<https://clarin.phonetik.uni-muenchen.de/BASWebServices/interface/ASR>

⁴<https://github.com/mozilla/DeepSpeech>

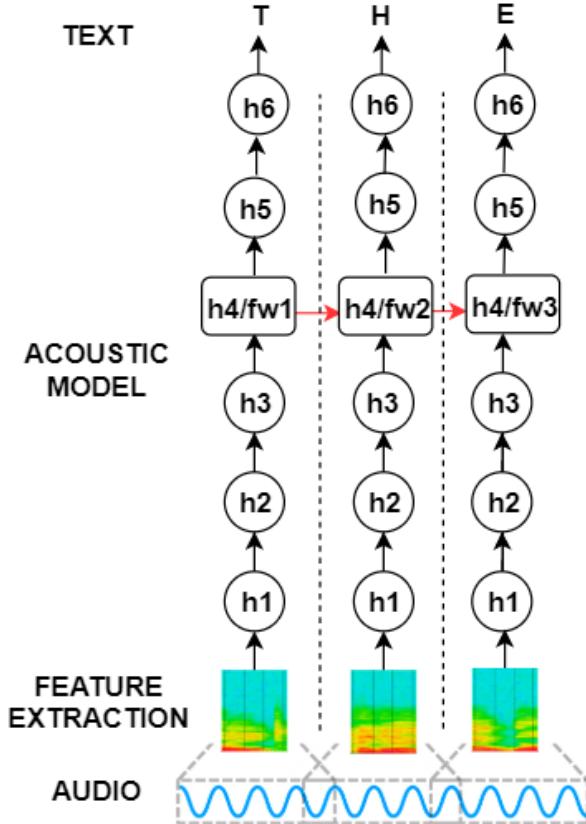


Figure 1: DeepSpeech architecture (adapted from *Mozilla Blog*⁵)

only one German model for any of these frameworks that is publicly available, which is the one by Milde and Köhn (2018) for Kaldi. Other German models, e.g. a Kaldi model from Fraunhofer IAIS (Stadtschnitzer et al., 2014), rely on in-house datasets and are not publicly available.

In this work, we focus on Mozilla’s DeepSpeech framework, as it is an end-to-end neural system that can be quite easily trained, unlike Kaldi, which requires more domain knowledge or wav2letter++, which is not yet widely tested by the community.

Mozilla DeepSpeech DeepSpeech (v0.1.0) was based on a TensorFlow (Abadi et al., 2016) implementation of Baidu’s end-to-end ASR architecture (Hannun et al., 2014). As it is under active development, the current architecture deviates from the original version quite a bit. In Figure 1, we give an overview of the architecture of version v0.5.0, which we also used for our experiments in this paper.⁶

DeepSpeech is a character-level, deep recurrent

neural network (RNN), which can be trained end-to-end using supervised learning.⁷ It extracts Mel-Frequency Cepstral Coefficients (Imai, 1983) as features and directly outputs the transcription, without the need for forced alignment on the input or any external source of knowledge like a Grapheme to Phoneme (G2P) converter. Overall, the network has six layers: the speech features are fed into three fully connected layers (dense), followed by a uni-directional RNN layer, then a fully connected layer (dense) and finally an output layer as shown in Figure 1. The RNN layer uses LSTM cells, and the hidden fully connected layers use a ReLU activation function. The network outputs a matrix of character probabilities, i.e. for each time step the system gives a probability for each character in the alphabet, which represents the likelihood of that character corresponding to the audio. Further, the Connectionist Temporal Classification (CTC) loss function (Graves et al., 2006) is used to maximize the probability of the correct transcription.

DeepSpeech comes with a pre-trained English model, but while Mozilla is collecting speech samples⁸ and is releasing training datasets in several languages (see paragraph on Mozilla Common Voice in Section 3), no official models other than English are provided. Users have reported on training models for French⁹ and Russian (Iakushkin et al., 2018), but the resulting models do not seem to be available.

3 Model Training

In this section, we describe in detail our setup for training the German model in order to ease subsequent attempts to train DeepSpeech models.

3.1 Datasets

To train the German Deep Speech model, we utilize the following publicly available datasets:

The **Voxforge**¹⁰ corpus, which is about 35 hours of German speech clips. Nearly 180 speakers have read aloud sentences from German Wikipedia, protocols from the European Parliament, and some individual commands. The clips vary in length, ranging from 5 to 7 seconds.

The **Tuda-De** (Milde and Köhn, 2018) corpus, is similar to Voxforge. It uses the same sources

⁵<https://hacks.mozilla.org/2018/09/speech-recognition-deepspeech>

⁶<https://github.com/mozilla/DeepSpeech/releases/tag/v0.5.0>

⁷<https://hacks.mozilla.org/2017/11/a-journey-to-10-word-error-rate/>

⁸<https://voice.mozilla.org/>

⁹<http://bit.ly/discourse-mozilla-org>

¹⁰<http://www.voxforge.org/home/forums/other-languages/german/open-speech-data-corpus-for-german>

Dataset	Size	Median Length	# Speakers	Condition	Type
Voxforge	35h	4.5s	180	noisy	read
Tuda-De	127h	7.4s	147	clean	read
Mozilla Common Voice	140h	3.7s	>1,000	noisy	read

Table 1: Overview of German datasets

(Wikipedia, parliament speeches, commands), but the recordings are under more controlled conditions. The final data was also curated “to reduce speaking errors and artefacts”. Each recording was made with 4 different microphones at the same time. This means that while the overall size of the dataset is larger than Voxforge and a model based on this dataset is supposed to be more robust, the actual amount of unique speech hours in both datasets are about the same.

The **Mozilla Common Voice** project¹¹ aims to make speech recognition open to everyone. The multilingual dataset currently covers 18 languages - including English, French, German, and Mandarin. The German corpus contains clips with lengths varying from 3 to 5 seconds. However, the corpus is recorded outside controlled conditions as per the comfort of the speaker. The utterances have background noise, and users have varied accents. Therefore we expect this dataset to be relatively challenging. Speakers in this dataset are relatively young, and the male/female ratio is about 5:1, which might result in a severe bias when trying to transfer the model.¹² The version used in our experiments has 140 hours of recordings, but as Mozilla aims at adding more recordings, there might already be a larger dataset available.

3.2 Preprocessing

DeepSpeech expects audio and transcription data to be prepared in a specific format so that they can be read directly by the input pipeline (see Figure 2 for an example). We cleaned the transcriptions by removing commas as well as punctuation and converting all transcriptions to lower case. We further ensured all audio clips are in .wav format. The pruned results were split into training (70%), validation (15%), and test data (15%).

For more details on data preprocessing parameters, we refer the reader to the code release.¹³

¹¹<https://voice.mozilla.org/de/datasets>

¹²Speaker Information is based on the self-reported statistics provided on the project homepage for each dataset.

¹³<https://github.com/AASHISHAG/deepspeech-german>

Hyperparameter	Value
Batch Size	24
Dropout	0.25
Learning Rate	0.0001

Table 2: Hyperparameters used in the experiments

3.3 Hyperparameter Setup

We searched for a good set of hyperparameters as shown in Figure 3. In the first iteration, we select learning rate and train batch-size and plot the graph showing the relationship of dropout and word-error rate, to determine the dropout with the lowest WER. We then used the best dropout (0.25) from the above iteration and kept the train batch size, to identify the best learning rate. Finally, we took the best dropout (0.25) and learning rate (0.0001) to determine the effect on batch size which shows that our initial choice of 24 was reasonable, even if somewhat better results seem possible using smaller batches.

Since Deep Speech employs early stopping, which stops the training of a neural network early before it overfits the training data, we did not experiment much with the number of epochs. The remaining hyperparameters were set to be the same as those pre-configured in Mozilla Deepspeech. The best results are obtained with the hyper-parameters mentioned in Table 2. We train the network using the Adam optimizer (Kingma and Ba, 2014).

Language Model We apply a probabilistic language model using KenLM toolkit (Heafield, 2011) to train a 3-gram model on the pre-processed corpus provided by Radeck-Arneth et al. (2015). It consists of eight million filtered sentences comprising 63.0% Wikipedia, 22.0% Europarl, and 14.6% crawled sentences. MaryTTS¹⁴ has been used to canonicalize the corpus, i.e. normalized to a form that is close to how a reader would speak the sentence, especially changing numbers, abbreviations, and dates. Additionally, punctuations were discarded, as it is usually also not pronounced. We

¹⁴<http://mary.dfki.de/>

wav_filename,wav_filesize,transcript
/voxforge/Manu-20100324-m25/wav/deM25-07.wav,156156,gartensaal wo der vater sie erwartete
/voxforge/anonymous-20130211-ehr/wav/de11-073.wav,156044,dadurch wird die zugriffzeit reduziert
/voxforge/Manu-20100324-m25/wav/deM25-18.wav,136732,ehrlich was sie eben dachten
/voxforge/b166er-20090404-hqj/wav/de4-27.wav,100044,welche obersten bundesorgane gibt es
/voxforge/AdrianTovar-20080727-xjh/wav/de11-145.wav,152962,zur bewährung ausgesetzt werden
/voxforge/Manu-20140331-m48/wav/deM48-25.wav,203584,er lief die treppe hinauf sie folgte langsam
/voxforge/Thomas-20120913-trl/wav/de11-009.wav,104836,die gegendarstellung ist offensichtlich
/voxforge/1337ad-20170321-ufl/wav/de3-39.wav,132044,sondern um die datenmenge

Figure 2: Screenshot of the input file format

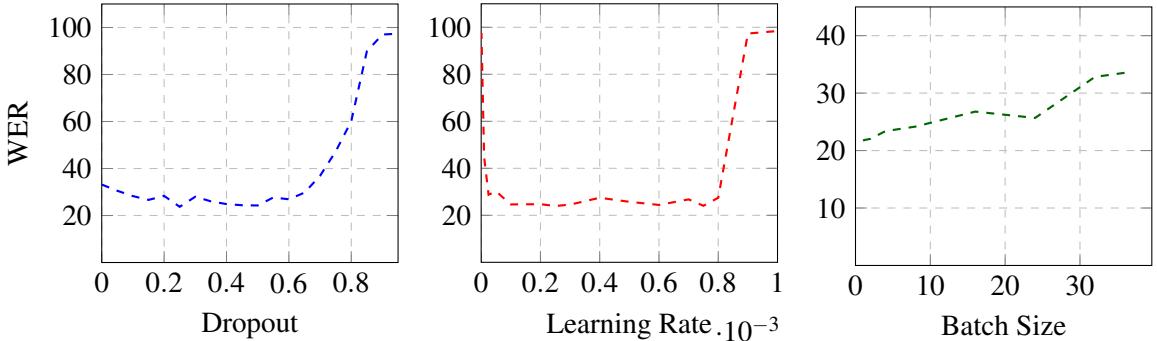


Figure 3: Hyperparameter search space

Dataset	WER
Mozilla	79.7
Voxforge	72.1
Tuda-De	26.8
Tuda-De + Mozilla	57.3
Tuda-De + Voxforge	15.1
Tuda-De + Voxforge + Mozilla	21.5

Table 3: German DeepSpeech results

used the unpruned Language Model that has a rather large vocabulary size of over 2 million types, but we expect pruning would only affect runtime, not recognition quality.

3.4 Server & Runtime

We trained and tested our models on a compute server having 56 Intel(R) Xeon(R) Gold 5120 CPUs @ 2.20GHz, 3 Nvidia Quadro RTX 6000 with 24GB of RAM each. Typical training time on a single dataset under this setup was in the range of 1 hour.

4 Results & Discussion

Table 3 shows the word error rates (WER) obtained when training and testing DeepSpeech on the available German datasets and their combinations. The best configuration in Milde and Köhn (2018) using only the Tuda-De corpus yields a WER of 28.96%.

Our model only trained on Tuda-De yields a comparable WER of 26.8%.

Results for the other datasets are much lower, but apparently combining several datasets improves the results. While the combination of Tuda and Mozilla yields a WER of 57.3%, the combination of Tuda, Voxforge, and Mozilla gives a WER of 21.5%. Combining the very similar Tuda-De and Voxforge yields a WER of 15.1%, which is a remarkable improvement over using only a single dataset. Note that this is the black-box performance, as we used DeepSpeech as is and only slightly tuned hyperparameters. See Section 6 for ideas on how to improve over these results.

To put our results into perspective, in Table 4, we present results in other languages for training different versions of the DeepSpeech architecture. Our best results are in the same range as for the other languages, but cross-dataset comparisons are hard to interpret. However, it is safe to say that training a DeepSpeech model can result in acceptable in-domain word error rates with considerably less training data than previously considered.

4.1 Influence of Training Size

Figure 4 depicts the relation between the amount of training data and its impact on the word-error-rate. To plot the learning curve, we split the training data into 10 subsets containing each 10% of the

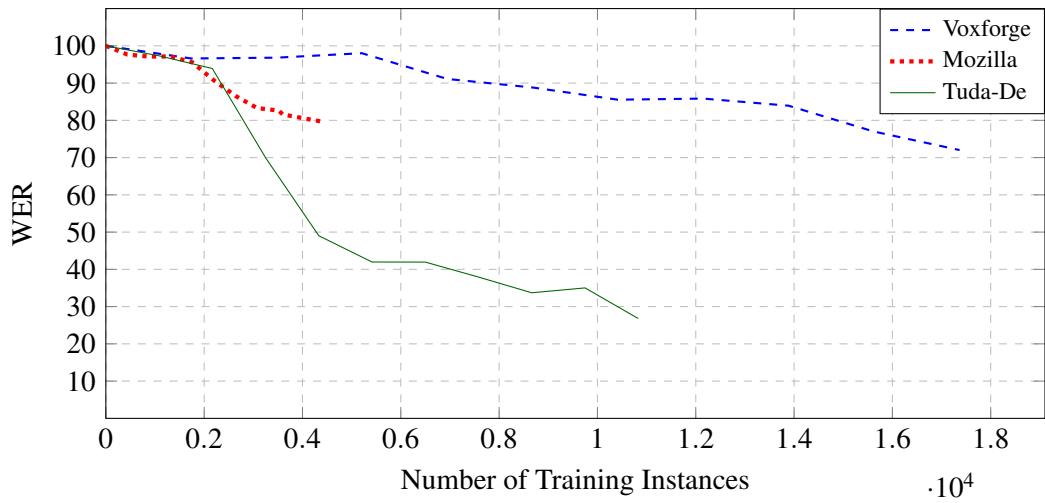


Figure 4: Learning curves for single datasets

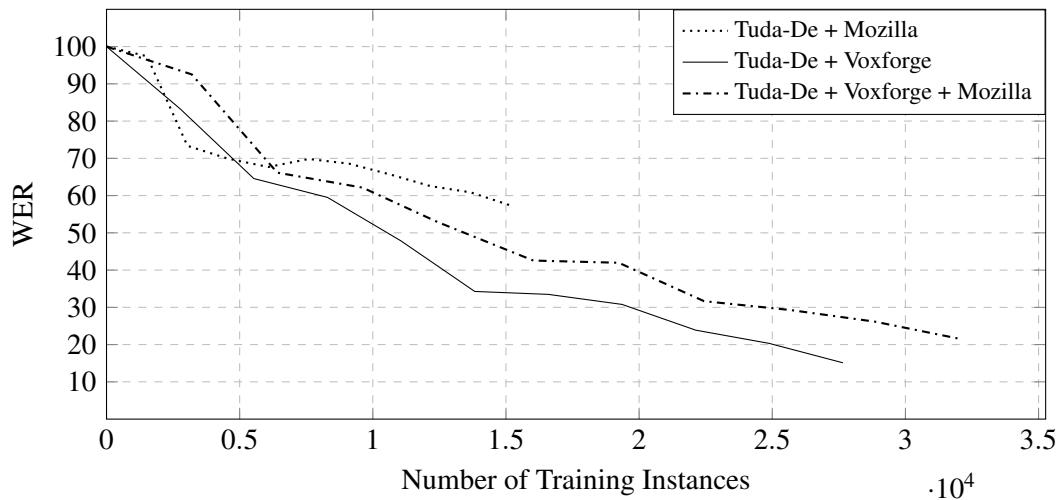


Figure 5: Learning curves when combining datasets

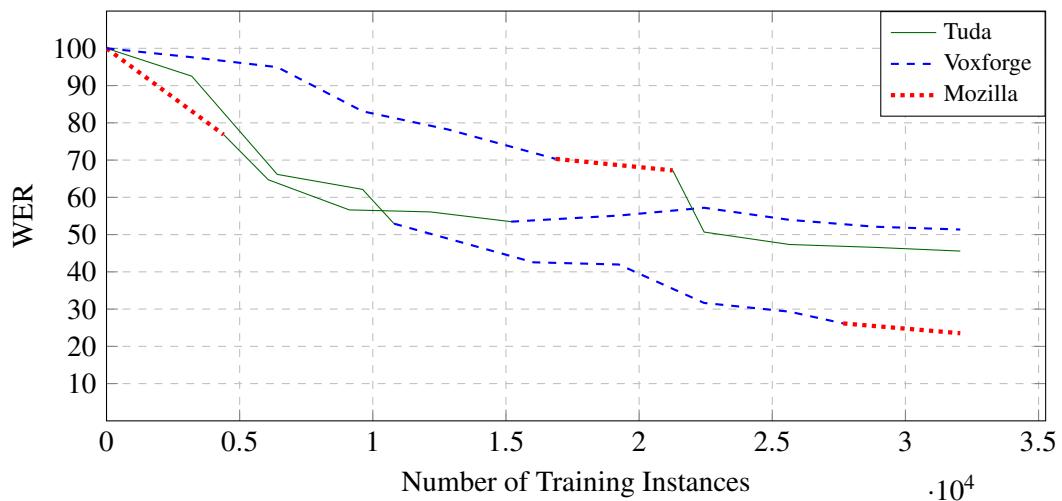


Figure 6: Order effects when combining datasets

Language	DeepSpeech version	Training Set	Size	Test Set	WER
English	Baidu (Hannun et al., 2014)	Switchboard Fisher WSJ Baidu	7,380h	Hub5 (LDC2002S23)	16.0
English	Mozilla v0.3.0	Switchboard Fisher LibriSpeech	3,260h	LibriSpeech (clean test)	11.0
English	Mozilla v0.5.0	Switchboard Fisher LibriSpeech	3,260h	LibriSpeech (clean test)	8.2
Russian	Mozilla v? (Iakushkin et al., 2018)	Yt-vad-1k Yt-vad-650-clean	1,650h	Voxforge (Russian)	18.0
German	Mozilla v0.5.0 (our)	Tuda-De + Voxforge	162h	Tuda-De + Voxforge (test)	15.1

Table 4: Comparison with previous results in other languages

training data. Then the model is trained on one subset and WER is calculated on a separate test dataset. Next, we introduce the new subset with more data, re-train the model, and compute its effect on the error rate. The model is trained on each subset for a maximum of 10 epochs and sometimes less when the model starts to overfit the training data, and early stopping is triggered. We observe that the rather noisy datasets Voxforge and Mozilla converge rather slowly, while the clean Tuda-De reaches much better results. This might also be a result of the different microphones that add increased robustness (not unlike other data augmentation strategies).

Figure 5 present the same learning curves when combining datasets showing that we can reach even better WER in this setting. Mixing the datasets seems to force the model to converge more quickly. However, combining the similar dataset Tuda-De and Voxforge yields a bit better performance than combining all three datasets.

We also tested against a mix of all datasets in combination, but add training data one dataset at a time. Thus, the order in which datasets are introduced into the training process might influence performance. Figure 6 shows the results for different order in which the datasets are introduced into the training process. Adding the noisy Mozilla dataset too early in the process seems to slow down convergence, while it adds a little bit of improved performance when added in the end.

4.2 Cross-dataset Performance

So far, we used training and testing data either from the same dataset or a mix of the available datasets,

Train	Test	WER
<i>Voxforge</i>		72.1
Tuda-De	Voxforge	96.8
Mozilla		73.1
Tuda-De, Mozilla		66.2
<i>Tuda-De</i>		26.8
Voxforge	Tuda-De	98.5
Mozilla		84.9
Voxforge, Mozilla		83.8
<i>Mozilla</i>		79.7
Tuda-De	Mozilla	94.8
Voxforge		87.1
Tuda-De, Voxforge		80.5

Table 5: Results across datasets

while of course keeping train and test data separate. To get a more realistic estimate of performance when used in a general setting, we assess cross-dataset performance, i.e. we train and develop on one or two datasets and test on a third one.

Table 5 shows the resulting word error rates. Apparently, the cross-domain results are much worse than in the in-domain setting in Table 3. For example, training on Mozilla or Voxforge and Mozilla and testing on Tuda-De yield unacceptable word error rates of 84.9 and 83.8 compared to 26.8 when training on Tuda-De. Interestingly, in this case, as we have seen already above, adding Voxforge in the mix does not help much, even if it is similar to Tuda-De. We see a similar picture for the other test datasets, transferring from a single dataset does not work at all, as in the training process the model is never forced to generalize beyond its properties.

However, training on the Tuda-De and Mozilla combination yields WER of 66.2 on Voxforge,

Model	WER	Example
<i>original</i>	-	<i>der bandbreitenverbrauch wird erheblich verringert</i>
Tuda-De	60	diese zeiten tonwoche erheblich verringert
Voxforge	80	zeiten epoche erheblich in
Tuda-De + Mozilla	160	es sind endete suche den ist es in
Tuda-De + Voxforge	60	der pen zeiten verprach wird erheblich verringert
Tuda-De + Voxforge + Mozilla	40	der bandbreiten verbrauch wird erheblich verringert
<i>original</i>	-	<i>ferner gibt es möglicherweise eine gewisse anonymität und sicherheit</i>
Tuda-De	78	weites mögliche welche in glichen unität und sicherheit
Voxforge	100	zitierweise sich entsichert
Tuda-De + Mozilla	100	hunde titisee gelten die die mitte zum
Tuda-De + Voxforge	44	den gibt es möglicherweise eine gewisse mietsicherheit
Tuda-De + Voxforge + Mozilla	11	er gibt es möglicherweise eine gewisse anonymität und sicherheit
<i>original</i>	-	<i>die einwilligung des schuldners war nicht erforderlich</i>
Tuda-De	100	ideen
Voxforge	86	die angebliche natacha vollich
Tuda-De + Mozilla	57	die einwilligung des schutzmacht erfordern
Tuda-De + Voxforge	86	die ein eigenes schuldnersicht erfordern
Tuda-De + Voxforge + Mozilla	43	die einigung des schuldner zwar nicht erforderlich
<i>original</i>	-	<i>die geschwindigkeit für die kunden kann erhöht werden</i>
Tuda-De	75	die geschwindigkeit und unterteilten
Voxforge	100	schinkelpreise
Tuda-De + Mozilla	88	wie die schmiede den trennendes
Tuda-De + Voxforge	38	die geschwindigkeit für die kunden kenterte
Tuda-De + Voxforge + Mozilla	0	die geschwindigkeit für die kunden kann erhöht werden
<i>original</i>	-	<i>mehrere arbeitgeberverbände sind zu einem dachverband zusammengeschlossen</i>
Tuda-De	114	der see aufweitungen des in einem tatorten samen erschossen
Voxforge	100	es recognitionszeichen
Tuda-De + Mozilla	100	in den sitzungen des entstandenen schaden
Tuda-De + Voxforge	29	mehrere arbeitgeberverbände sind zu einem tachodata geschlossen
Tuda-De + Voxforge + Mozilla	14	der arbeitgeberverbände sind zu einem dachverband zusammengeschlossen

Table 6: Recognition results on random Voxforge test instances

which is even lower than using the training portion of Voxforge (which yields 72.1). Thus forcing the model to generalize over topics, recording conditions, speakers, etc. seem to be a crucial point.

5 Error Analysis

Table 6 shows the recognition results on randomly selected test instances from the Voxforge dataset. The models trained on only one dataset are surprisingly bad, resulting in rather poetic utterances that sometimes are quite far from the expected source. An example is the Tuda-De model recognizing *tatorten samen erschossen* instead of *dachverband zusammengeschlossen*.

As is to be expected for German, compounds are especially challenging as exemplified by *bandbreitenverbrauch* that is recognized as *bandbreiten verbrauch* or even *pen zeiten verprach*, where *verprach* is probably only in the language model as a common misspelling of *versprach*.

The models often fail in interesting ways, e.g. all models sometimes return very short results like *schinkelpreise* that should actually have low prob-

ability. We currently have no explanation for this behaviour and need to explore the issue further.

In cases like *des schuldners war* being recognized as *des schuldner zwar*, the phonetic ambiguity should have been resolved by a better language model.

6 Summary

In this paper, we presented the first results on building a German speech recognition model using Mozilla Deep Speech. Our best performing model reaches an in-domain WER of 15.1%, which is in line with the performance for other languages using the DeepSpeech framework. Our results thus support the idea that Mozilla Deep Speech can be easily transferred to new languages. Learning curve experiments highlight the importance of the amount of training data, but also quite strong order effects when mixing the datasets.

We publish our trained model along with configuration data for all our experiments in order to enable replicating all results. The model can easily be re-trained and optimised on new datasets by

referring the code-release.¹⁵ No specific hardware is required to run the trained model, and it works even on a normal desktop computer or laptop.

Future Work Our experiments only scratch the surface of possible approaches, and our analysis recommends several avenues for further exploration.

We mainly treated DeepSpeech as a black-box and only performed a light hyper-parameter search. The model can probably still be fine-tuned by exploring other hyper-parameters. We also did not experiment much with the language model, but used a simple 3-gram model.

Since the amount of publicly available training data is limited, it could be interesting to consider data augmentation strategies.¹⁶ Another approach to improve recognition quality could be to use transfer learning by taking an English model (pre-trained with the larger English datasets) and re-training with the German data (Kunze et al., 2017; Bansal et al., 2018). In the light of recent discussions on the CO₂ footprint of training deep learning models (Strubell et al., 2019), using re-training and providing trained models is desirable. Additionally, more research is needed to find neural architectures that perform equally well, but require less compute.

Finally, the training process described here could be easily used to train speech recognition models for other languages, where currently no pre-trained models are available.

Acknowledgments

We want to thank Andrea Horbach for her many helpful comments that significantly improved the paper. We also thank the developers at Mozilla DeepSpeech, who provided insight and expertise that greatly assisted the research.

References

- [Abadi et al.2016] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation OSDI 16*, pages 265–283.

- [Bansal et al.2018] Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater.

¹⁵<https://github.com/AASHISHAG/deepspeech-german>

¹⁶<https://ai.googleblog.com/2019/04/specaugment-new-data-augmentation.html>

2018. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. *CoRR*, abs/1809.01431.

- [Gaida et al.2014] Christian Gaida, Patrick Lange, Rico Petrick, Patrick Proba, Ahmed Malatawy, and David Suendermann-Oeft. 2014. Comparing open-source speech recognition toolkits.

- [Graves et al.2006] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. volume 2006, pages 369–376, 01.

- [Hannun et al.2014] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. 2014. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567.

- [He et al.2018] Yanzhang He, Tara N. Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, Qiao Liang, Deepti Bhatia, Yuan Shangguan, Bo Li, Golan Pundak, Khe Chai Sim, Tom Bagby, Shuo-Yiin Chang, Kanishka Rao, and Alexander Gruenstein. 2018. Streaming end-to-end speech recognition for mobile devices. *CoRR*, abs/1811.06621.

- [Heafield2011] Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland.

- [Iakushkin et al.2018] Oleg Iakushkin, George Fedoseev, Anna S. Shaleva, Alexander Degtyarev, and Olga S. Sedova. 2018. Russian-language speech recognition system based on deepspeech. In *Proceedings of the VIII International Conference on Distributed Computing and Grid-technologies in Science and Education (GRID 2018)*.

- [Imai1983] Satoshi Imai. 1983. Cepstral analysis synthesis on the mel frequency scale. In *ICASSP'83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 8, pages 93–96. IEEE.

- [Kingma and Ba2014] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, Dec.

- [Krstovski et al.2008] Kriste Krstovski, Michael Decerbo, Rohit Prasad, David Stallard, Shirin Saleem, and Premkumar Natarajan. 2008. A wearable headset speech-to-speech translation system. In *Proceedings of the ACL-08: HLT Workshop on Mobile Language Processing*, pages 10–12, Columbus, Ohio, June. Association for Computational Linguistics.

[Kunze et al.2017] Julius Kunze, Louis Kirsch, Ilia Kurenkov, Andreas Krug, Jens Johannsmeier, and Sebastian Stober. 2017. Transfer learning for speech recognition on a budget. *CoRR*, abs/1706.00290.

[Li et al.2017] Bo Li, Tara Sainath, Arun Narayanan, Joe Caroselli, Michiel Bacchiani, Ananya Misra, Izhak Shafran, Hasim Sak, Golan Pundak, Kean Chin, Khe Chai Sim, Ron J. Weiss, Kevin Wilson, Ehsan Variani, Chanwoo Kim, Olivier Siohan, Mitchel Weintraub, Erik McDermott, Rick Rose, and Matt Shannon. 2017. Acoustic modeling for google home.

[Liao et al.2013] Hank Liao, Erik McDermott, and Andrew W. Senior. 2013. Large scale deep neural network acoustic modeling with semi-supervised training data for youtube video transcription. In *ASRU*, pages 368–373. IEEE.

[Milde and Köhn2018] Benjamin Milde and Arne Köhn. 2018. Open source automatic speech recognition for german. *CoRR*, abs/1807.10311.

[Povey et al.2011] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, December.

[Pratap et al.2018] Vineel Pratap, Awni Hannun, Qiantong Xu, Jeff Cai, Jacob Kahn, Gabriel Synnaeve, Vitaliy Liptchinsky, and Ronan Collobert. 2018. wav2letter++: The fastest open-source speech recognition system. *CoRR*, abs/1812.07625.

[Radeck-Arneth et al.2015] Stephan Radeck-Arneth, Benjamin Milde, Arvid Lange, Evandro Gouvêa, Stefan Radomski, Max Mühlhäuser, and Chris Biemann. 2015. Open source german distant speech recognition: Corpus and acoustic model. In *Text, Speech, and Dialogue*, pages 480–488, Cham.

[Saon et al.2017] George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, Bergul Roomi, and Phil Hall. 2017. English conversational telephone speech recognition by humans and machines. *CoRR*, abs/1703.02136.

[Stadtschnitzer et al.2014] Michael Stadtschnitzer, Jochen Schwenninger, Daniel Stein, and Joachim Koehler. 2014. Exploiting the large-scale German Broadcast Corpus to boost the Fraunhofer IAIS Speech Recognition System. In *Proceedings of LREC 2014*, pages 3887–3890, Reykjavik, Iceland.

[Strubell et al.2019] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. In *Proceedings of ACL*.

Label Propagation of Polarity Lexica on Word Vectors

Harald Koppen

cellent GmbH

Lehrer-Wirth-Str. 2

81829 München

haraldkoppen@gmail.com

Ritavan

Surplus Digital GmbH

Georg-Muche-Straße 5

80807 München

ritavan.saice@gmail.com

Abstract

The Semi-supervised learning (SSL) is an important research area in machine learning where both labeled and unlabeled data is used to build a model. One of the big advantages of semi-supervised methods is that they are transparent and easy to comprehend for humans, unlike most deep learning techniques which are black box. In this paper, we design a graph-based semi-supervised learning framework to detect sentiment polarity in word vectors trained on a German corpus. We study theoretical aspects of the task, empirically analyze a seminal label propagation algorithm (Zhu and Ghahramani, 2002) and suggest variants to improve classification performance. Additionally, we review the literature of graph construction for SSL and propose new methods to avoid hubs, i.e., vertices of high degree, which are harmful as outlined by Ozaki et al. (2011).

1 Introduction

Among the most ubiquitous techniques for label enrichment and transfer learning in sentiment analysis, in particular for classification tasks, are sentiment lexica and word vectors. The use of such lexica is a classical approach which has been used for several decades before the advent of deep learning (Taboada et al., 2011). The training of word vectors from large unlabeled text corpora is a comparatively more recent method dating back to the seminal paper by Mikolov et al. (2013).

For sentiment analysis, it is common to focus on supervised methods (Gamon, 2004; Matsumoto et al., 2005; Pang et al., 2002; dos Santos and Gatti, 2014). Usually, large unlabeled text corpora are easily available, whereas labeled lexica are harder to come by and often involve exorbitant labeling

costs. Thus, given an unlabeled dataset at the outset, this approach is expensive as it takes both time and labor to annotate a sufficiently large training set. Typically, word vectors have a vocabulary of size $O(10^6)$ (Mikolov et al., 2013) while lexica contain $O(10^4)$ (Waltinger, 2010a) words, thus resulting in a poor ratio of labeled to unlabeled points.

In recent years, semi-supervised learning (SSL) methods, particularly graph-based approaches based on label propagation (Zhu and Ghahramani, 2002) attracted attention (Goldberg and Zhu, 2006; Rao and Ravichandran, 2009; Ren et al., 2012). As a consequence, graph construction for these methods emerged as a relevant field of study (Ozaki et al., 2011; de Sousa et al., 2013; Vega-Oliveros et al., 2014) as well as approaches minimizing a cost function derived from such a graph (Ravi and Diao, 2016). Note that label propagation and its variations are equivalent to certain minimization problems (Bengio et al., 2006).

Giulianelli (2017) used SSL on a word embedding obtained via a layer of a long short term memory (LSTM) recurrent network instead of using word vectors. However, training an LSTM is a supervised task, i.e. the method requires a large amount of labeled data in the first place, which defeats the purpose and is going against the main motivation behind SSL techniques.

A major challenge with high dimensional data is the curse of dimensionality, a well-known phenomenon particularly affecting methods based on nearest neighbour graphs. Radovanović et al. (2010) and subsequently Ozaki et al. (2011) showed that hubs, i.e. vertices of high degree, have a negative effect on classification results due to the fact that they are among the nearest neighbours of a large subset of the dataset.

We introduce the k nearest neighbor (k NN) graph, consider different variants of it and propose a trimming and a normalization procedure in order to combat hubs.

2 Contributions

To the best of our knowledge, there is no previous work carrying out a detailed theoretical and empirical study of SSL as described above, that is label propagation of a German sentiment lexicon on word vectors trained on a German corpus.

Our contributions are as follows:

- A study of theoretical challenges of label propagation on a polarity lexicon of word vectors.
- Benchmarking the performance of label propagation on different word vector models of varying dimensionality, including contextual language models.
- Extensive experiments to study the performance of label propagation empirically with a variety of parameter configurations and graph construction techniques.
- Proposition of 2 new methods to avoid the negative effect of hubs during label propagation.

The rest of this paper is organized in the following way. We introduce the SSL setting, label propagation and its problems in section 3. Our new methods for graph regularization are explained in section 4. Further motivation, analysis of the dataset used, the set-up and the results of our experiments are given in section 5. We conclude with section 6.

3 Graph-based SSL

We begin with a definition of SSL, then define the similarity function. Afterwards, we move on to graph construction and label propagation before discussing the challenges faced by these methods.

3.1 Similarity and Semi-Supervised Learning

Assuming the data is already given as a finite set of points in \mathbb{R}^d , $d \in \mathbb{N}$, let $l \in \mathbb{N}$ denote the number of labeled points, $u \in \mathbb{N}$ the number of unlabeled points and $n = l + u$ the total number of points. We are considering $\mathcal{L} = \{x_1, \dots, x_l\} \subseteq \mathbb{R}^d$, the set of labeled points, and $\mathcal{U} = \{x_{l+1}, \dots, x_n\} \subseteq \mathbb{R}^d$, the set of unlabeled points, where $x_i \neq x_j$ for every $i \neq j$, i.e. the points are pairwise distinct. The label of x_i is denoted by $y_i \in \{0, \dots, \rho\}$, $\rho \in \mathbb{N}$. In this paper, we study binary classification, i.e. $\rho = 1$. Given $\{y_1, \dots, y_l\}$, the goal of SSL is to predict $\{y_{l+1}, \dots, y_n\}$ as accurately as possible.

The similarity function is a map

$$\sigma : \mathcal{L} \cup \mathcal{U} \times \mathcal{L} \cup \mathcal{U} \longrightarrow \mathbb{R}^+, (x, x') \mapsto \sigma(x, x'),$$

for instance

$$\sigma_\gamma(x, x') = f_\gamma(x - x'),$$

where

$$f_\gamma : \mathbb{R}^d \longrightarrow \mathbb{R}^+, x \mapsto e^{-\frac{\|x\|_2^2}{2\gamma^2}}$$

denotes the radial basis function and $\gamma > 0$.

Another example makes use of the k nearest neighbors of x in $\mathcal{L} \cup \mathcal{U}$, defined as follows. Let $k \in \{1, \dots, n-1\}$, $x \in \mathbb{R}^d$ and $x_{(1)}, \dots, x_{(n)}$ be a reordering of $\mathcal{L} \cup \mathcal{U}$ such that

$$\|x_{(1)} - x\|_2 \leq \dots \leq \|x_{(n)} - x\|_2.$$

Then the k nearest neighbors of x in $\mathcal{L} \cup \mathcal{U}$ are

$$k\text{NN}(x, \mathcal{L} \cup \mathcal{U}) = \{x_{(1)}, \dots, x_{(k)}\}.$$

Now, we can define

$$\sigma_k(x, x') = \begin{cases} 1 & x' \in k\text{NN}(x, \mathcal{L} \cup \mathcal{U}) \\ 0 & \text{otherwise} \end{cases}. \quad (*)$$

Note that for every $i \in \{1, \dots, n\}$ and every k we have that $x_i \in k\text{NN}(x_i, \mathcal{L} \cup \mathcal{U})$. To avoid this, one can define $k\text{NN}(x, \mathcal{L} \cup \mathcal{U}) = \{x_{(2)}, \dots, x_{(k+1)}\}$. Including the distance of x and x' is possible by using

$$\sigma_{k,\gamma}(x, x') = \begin{cases} f_\gamma(x - x') & x' \in k\text{NN}(x, \mathcal{L} \cup \mathcal{U}) \\ 0 & \text{otherwise} \end{cases}.$$

3.2 Construction of the Underlying Graph

The vertices of the underlying graph are given by $\mathcal{L} \cup \mathcal{U}$. Consider the adjacency matrix $A \in \mathbb{R}^{n \times n}$ which is derived from the similarity matrix defined as $W = (\sigma(x_i, x_j))_{1 \leq i, j \leq n}$.

The easiest choice for A is W itself, where $\sigma = \sigma_\gamma$ yields a dense, undirected and weighted graph. As A is usually heavily involved in the classification of \mathcal{U} it is desirable to use a sparse matrix to save computation time. In particular, a sparse adjacency matrix results in higher classification accuracy as noise and spurious relationships are reduced (Zhu, 2008; Ozaki et al., 2011).

Taking $\sigma = \sigma_k$ leads to a sparse, directed and unweighted graph, $\sigma = \sigma_{k,\gamma}$ to a sparse, directed and weighted graph known as a k NN graph. Usually, it is transformed into an undirected graph by choosing the adjacency matrix

$$W_{\max} = (\max(\sigma(x_i, x_j), \sigma(x_j, x_i)))_{1 \leq i, j \leq n}.$$

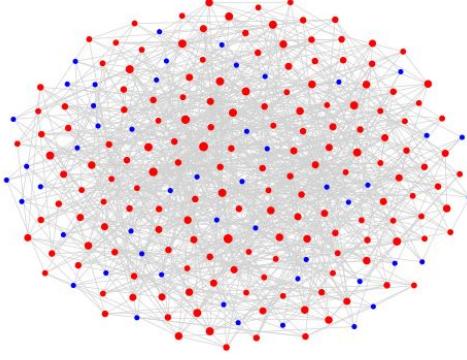


Figure 1: 5NN graph on UCI glass data set, where vertices with degree larger than 5 are drawn red and accordingly bigger.

Ozaki et al. (2011) study the mutual k NN graph which is given by the adjacency matrix

$$W_{\min} = (\min(\sigma(x_i, x_j)), \sigma(x_j, x_i)))_{1 \leq i, j \leq n}.$$

Note that W_{\max} and particularly W_{\min} may yield to a disconnected graph, harming the classification if there are connected components with few or absolutely no labeled points.

In any case, the use of σ_k results in self-loops as $\sigma_k(x_i, x_i) = 1$ for every $i \in \{1, \dots, n\}$. These can be removed by using the modified version of k NN mentioned below (*). Note further that $\sigma_{k,\gamma}(x, x) = 0$ for every $x \in \mathbb{R}^d$, i.e. for fixed $i \in \{1, \dots, n\}$ there are not k , but $k - 1$ non-zero entries in $(\sigma_{k,\gamma}(x_i, x_j))_{1 \leq j \leq n}$. Again, the modified version of k NN prevents this behaviour.

3.3 Label Propagation

For the moment, let us assume $y_i \in \{-1, 1\}$, i.e. we replace the label 0 by -1 . Given an adjacency matrix A , the algorithm is given as follows (Bengio et al., 2006).

Algorithm 1 Label Propagation

Compute A

Compute diagonal D by $D_{ii} \leftarrow \sum_{j=1}^n A_{ij}$

Initialize $Y^{(0)} \leftarrow (y_1, \dots, y_l, 0, \dots, 0)$

Iterate

1. $Y^{(t+1)} \leftarrow D^{-1} A Y^{(t)}$

2. $Y_i^{(t+1)} \leftarrow y_i$ for $1 \leq i \leq l$

until convergence criterion is satisfied

Denote the result by $Y^{(\infty)}$

Set $y_i = \text{sgn}(Y_i^{(\infty)})$

Consequently, the algorithm propagates the information along the edges of the underlying graph,

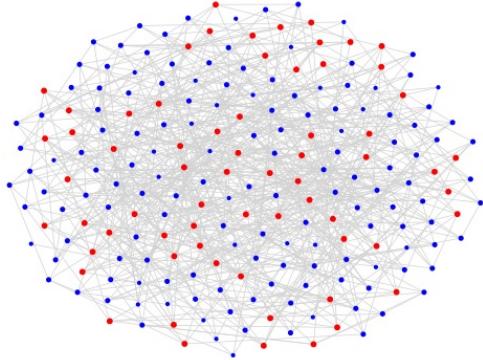


Figure 2: Trimmed version of Figure 1, $\alpha = 5$.

typically until an equilibrium state is reached. The nodes initially labeled serve as the source of information.

A classical assumption in SSL is the *cluster assumption*: if points are in the same cluster, they are likely to be of the same class (Chapelle et al., 2009). Of course, for high-dimensional data, it is hard to check if this assumption is fulfilled, especially given that only a small proportion of the data is labeled. Strictly speaking, this problem should be overcome by the word embedding algorithm, not the label propagation algorithm.

4 Improvements to the Graph

Let us now consider the undirected unweighted k NN graph without self-loops, that is, the graph $G = (\mathcal{L} \cup \mathcal{U}, W_{\max})$ with similarity function σ_k using the modified version of k NN.

4.1 ε -Sparsification

Let $\varepsilon > 0$. The ε -sparsification of G is the graph G^ε which is obtained by deleting every edge $\{x_i, x_j\}$ in G where $\|x_i - x_j\|_2 > \varepsilon$. Therefore, using G^ε instead of G reduces the influence of outliers on the classification.

4.2 Edge Normalization

Firstly, we propose to transform G into a weighted graph G^n by performing *edge normalization*, i.e. by assigning every edge $\{u, v\}$ in G the weight

$$w_{u,v} = (\deg_G(u) + \deg_G(v))^{-1}.$$

Note that $w_{u,v}$ is small if u and v have high degree and vice versa, thus counterbalancing the high amount of edges between vertices with high degree.

Let $N_G(u)$ denote the set of neighbors of u in G .

For every vertex $u \in G$, we have

$$\begin{aligned} 0 &< \sum_{v \in N_G(u)} w_{u,v} \\ &\leq \sum_{v \in N_G(u)} (\deg_G(u) + \min_{v \in N_G(u)} \deg_G(v))^{-1} \\ &= \frac{\deg_G(u)}{\deg_G(u) + \min_{v \in N_G(u)} \deg_G(v)} < 1, \end{aligned}$$

i.e. the weighted degree in G^n is concentrated on the unit interval $(0, 1)$.

4.3 Edge Trimming

Secondly, one can apply *edge trimming* to G in order to obtain G' , i.e. one deletes edges in G by the procedure given as follows:

1. Choose a threshold $\alpha \geq k$ and define $\mathcal{H} = \{u \in G \mid \deg_G(u) > \alpha\}$
2. For every u in \mathcal{H} , let $v_1^u, \dots, v_{\deg_G(u)}^u$ be a reordering of $N_G(u)$ such that $\deg_G(v_1^u) \geq \dots \geq \deg_G(v_{\deg_G(u)}^u)$
3. For every u in \mathcal{H} remove the edges $\{u, v_1^u\}, \dots, \{u, v_l^u\}$ from G (if possible) where $l = \deg_G(u) - \lfloor k \log_k(\deg_G(u)) \rfloor$

Figures 1 and 2 illustrate the usefulness of trimming for the regularization of k NN graphs using the UCI glass data set (Dua and Graff, 2017).

4.4 Computational Efficiency

Jebara et al. (2009) and Ozaki et al. (2011) reported that so-called b -matching graphs, a special case of b -regular graphs, achieve higher classification accuracy than k NN graphs. However, constructing the b -matching graph takes $O(bn^3)$ time (Huang and Jebara, 2007) which is too long to be useful in practice when having large amounts of data. Therefore, regularizing G within a reasonable amount of time is desirable.

Fredman and Tarjan (1987) showed that the complexity of building G is $O(n^2 + kn \log n)$. As the number of edges in G is bounded by kn , the construction time of G^e , G^n or G' given G is $O(kn)$. Hence the overall construction time is dominated by the term $O(n^2 + kn \log n)$.

Note that approximate k NN graphs can be constructed in $O(kn)$ time (Beygelzimer et al., 2006; Chen et al., 2009; Ram et al., 2009; Tabei et al., 2010). Combining these with the modifications discussed above yields a graph construction algorithm having time complexity $O(kn)$.

	NN	VV	AD	Other	Total
polar	4028	1810	3621	102	9561
neutral	642	253	254	61	1210

Table 1: Absolute frequencies of POS-tags among the labeled word vectors.

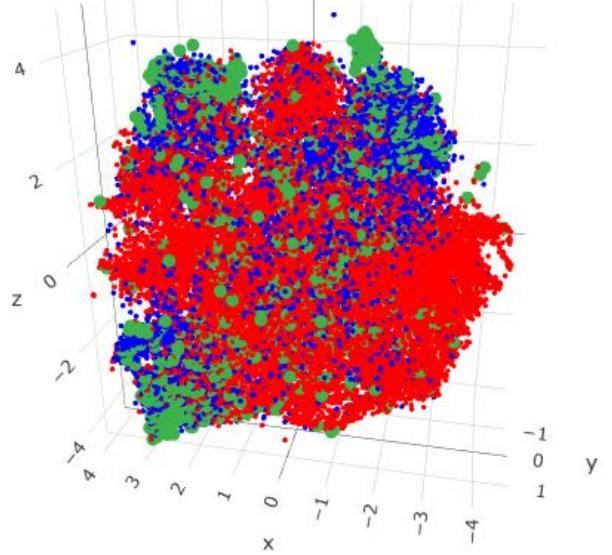


Figure 3: T-SNE plot (perplexity = 40) of the neutral (green), polar (blue) and unlabeled (red) word vectors. For sake of clarity, only 20000 red dots are shown.

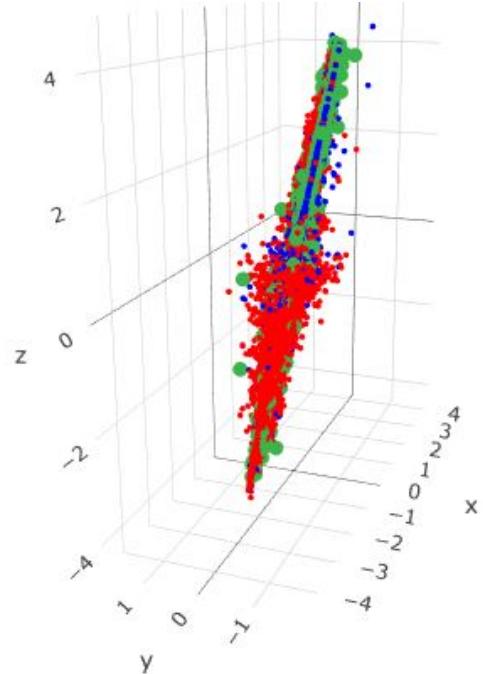


Figure 4: Figure 3 rotated around the x -axis.

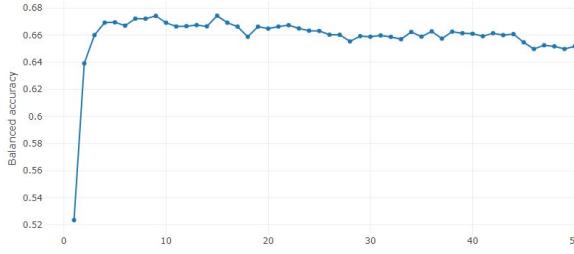


Figure 5: Balanced accuracy for label propagation on the k NN graph.

5 Experiments

In this paper, we use the lemmatized sentiment lexica introduced in (Waltinger, 2010a) and (Waltinger, 2010b) and label propagation for word-level polarity lexicon expansion.

We compare the label propagation algorithm given above on various graphs in a sentiment polarity detection task. More precisely, we consider G as in section 4 and its modifications as well as the undirected weighted k NN graph without self-loops, i.e. the graph $G_\gamma = (\mathcal{L} \cup \mathcal{U}, W_{\max})$ with similarity function $\sigma_{k,\gamma}$ using the modified version of k NN. The convergence criterion is given by $\|Y^{(t+1)} - Y^{(t)}\|_1 < 0.001$.

5.1 Dataset and Resources

As there was no public FastText model (Bojanowski et al., 2016; Joulin et al., 2016) trained on a proprietary German news corpus, we trained our own model. The resulting vocabulary size was 196972 word vectors of dimension 60. The reason for choosing FastText was the ability of the trained model to deal with out of vocabulary (OOV) words, as it is using subword character information.

The labeled word vectors are given by the lemmatized dictionaries used in (Waltinger, 2010a; Waltinger, 2010b). We assign the label 1 to the words annotated positive or negative, i.e. polar, and 0 to the words annotated neutral, where we removed the digits and the punctuation symbols from the neutral dictionary.

We prefer this lexicon over SentiWS (Remus et al., 2010) and PolArt (Klenner et al., 2009) as it is the largest one - 10771 words compared to approximately 3450 and 9380, respectively. Furthermore, SentiWS measures sentiment using the full interval $[-1, 1]$, i.e. first, one has to categorise the sentiment value before one can apply label propagation.

In particular, polarity is sparsely embedded in

language, i.e. a model accurately determining polarity can be used to extend sentiment dictionaries.

We choose to learn neutral vs. polar as the usually treated three-way case is significantly harder on word-level. For instance, the sentiment of ‘rise’ is polar, but the precise value depends heavily on the context (e.g. compare *wealth is rising* and *poverty is rising*).

5.2 Description of Dataset

The ability to embed OOV words is an integral part of our method as the labeled words are not necessarily contained in the corpus mentioned above. Figures 3 and 4 show a three-dimensional t-SNE plot (Maaten and Hinton, 2008; Van Der Maaten, 2014) of the word vectors, indicating that the dataset is lying on a low-dimensional manifold.

In total, we have 9561 data points with label 1 and 1210 data points with label 0. Only 163 ($\approx 1.5\%$) of these words have a Part-Of-Speech-tag (POS-tag) that is not noun (NN), verb (VV) or adjective (AD) (see Table 1). Therefore we only consider unlabeled words whose POS-tag is one of these three, reducing the amount of unlabeled points to 85759.

We randomly draw a test set of 3000 words from the set of unlabeled points. The test set is labeled by one of the authors. 362 words ($\approx 12.1\%$) were assigned “polar”.

Comparing with an independent annotator, we have an inter-annotator agreement of Cohen’s $\kappa = 0.4682$ showing that word-level sentiment analysis is a very hard to perform task, even for humans. Consequently, one cannot expect a model predicting sentiment to be performing as well as prediction models in different areas of Machine Learning.

This is probably due to the fact that sentiment is subjective and thus influenced by the emotional association of words to experiences of the individual annotator. There are even studies that suggests that the voice and audio signal is as important as the text for semantic purposes. A more fundamental fact is that sentiment in human language is better identified given the context, thus rendering the analysis of word-level sentiment even harder.

5.3 Dimensionality and Information Content of the Embedded Data

Given the ever increasing dimensionality of embeddings, from about 300 in the early Word2Vec models to more than 3000 in the most recent contextualized embeddings like ELMo (Peters et al.,

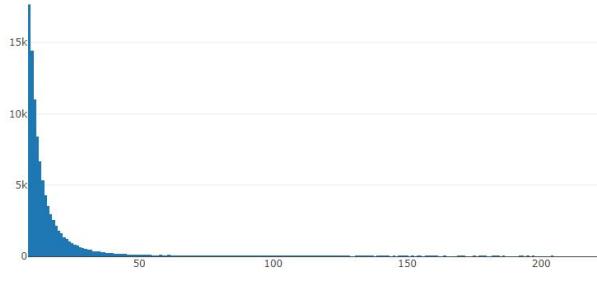


Figure 6: Distribution of degree in G .

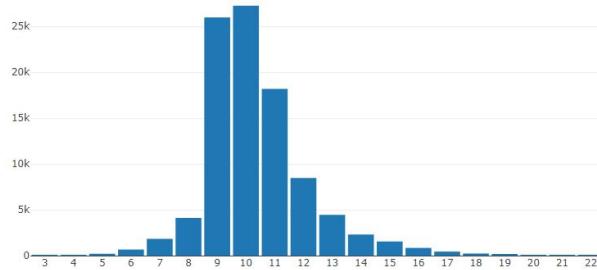


Figure 7: Distribution of degree in G^t , $\alpha = 13$.

2018), we study the cumulative explained variation of the word embeddings given by Principal Component Analysis (PCA) (Pearson, 1901) and examine it for decreasing dimension of the target space.

In every case, there is a decay starting out slowly, followed by a very sharp drop suggesting that most of the critical information content of the given word embedding is lying on a low-dimensional manifold.

5.4 Class Balancing and Parameters

Instead of transforming our labels to -1 and 1 (recall section 3.3), we normalized the labels by class size, i.e. we used $-1/1210$ for the neutral

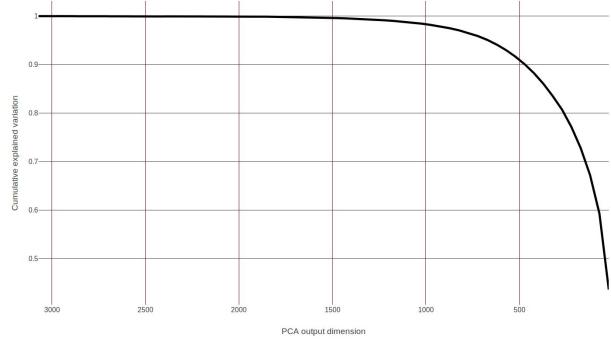


Figure 9: Cumulative explained variation of PCA on our data embedded using ELMo.

words and $1/9561$ for the polar words.

For all experiments, we use the same 9NN graph G as $k = 9$ maximizes the balanced accuracy (see Figure 5). Given the near linear time complexity of modifying G , we obtained an optimal parameter configuration using binary search.

Note that word vector models trained on a very large vocabulary, OOV words almost never occur. Hence, we also compare our self-trained embedding with different pre-trained ones.

5.5 Comparison of Word Vector Embeddings and Classification Results

Training a word vector model on the corpus at hand is usually an expensive and rewarding step at the same time. We compare our model with three pre-trained word embeddings:

- FastText,
- ELMo and
- GloVe (Pennington et al., 2014).

Note that ELMo is a contextualized representation model embedding a word within its sentence. As we are working on word-level, each sentence is the word itself. The results are shown in table 2.

Despite being the lowest-dimensional, our self-trained model captures the nuances of our corpus better than the other pretrained models. Further, we can see that ELMo, one of the most recent contextualized word embedding models, clearly outperforms FastText and GloVe, whereas the latter two roughly score the same.

Table 3 shows the result for G_γ , G and its modifications using the parameters maximizing the F1 score. We can see that G_γ is performing worse than G and its modifications. In particular, removing hubs via edge normalization or trimming is improving classification performance.

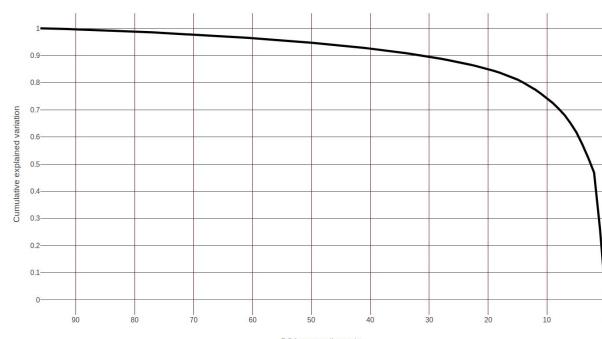


Figure 8: Cumulative explained variation of PCA on our data embedded using GloVe.

Embedding	Dimension	F1	Recall	Precision	Bal. Acc.
FastText (self-trained)	60	0.3405	0.6381	0.2322	0.6743
GloVe	96	0.2084	0.4199	0.1386	0.5308
FastText (pre-trained)	300	0.2016	0.2376	0.1752	0.5420
ELMo	3072	0.2602	0.6492	0.1627	0.5954

Table 2: F1 score and balanced accuracy for G with different word embeddings to transform our data into high-dimensional vectors.

Underlying Graph	F1	Recall	Precision	Bal. Acc.
$G_\gamma, \gamma = 14$	0.3317	0.6630	0.2212	0.6713
G	0.3405	0.6381	0.2322	0.6743
$G^\epsilon, \epsilon = 110$	0.3410	0.6381	0.2326	0.6746
G^n	0.3437	0.6575	0.2326	0.6799
$(G^\epsilon)^n, \epsilon = 110$	0.3449	0.6602	0.2334	0.6813
$G^t, \alpha = 13$	0.3428	0.6685	0.2305	0.6811
$(G^\epsilon)^t, \epsilon = 120, \alpha = 12$	0.3437	0.6740	0.2306	0.6827

Table 3: F1 score and balanced accuracy for G_γ , G and G with different combinations of the modifications discussed in section 4. $(G^\epsilon)^n$ indicates that edge normalization was applied after ϵ -sparsification.

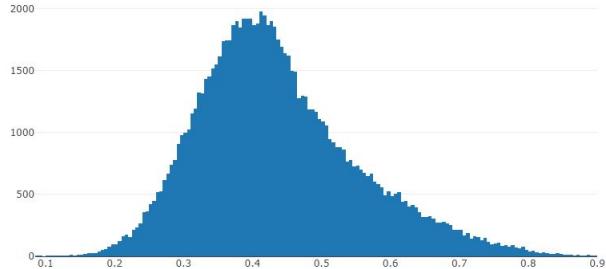


Figure 10: Distribution of weighted degree in G^n .

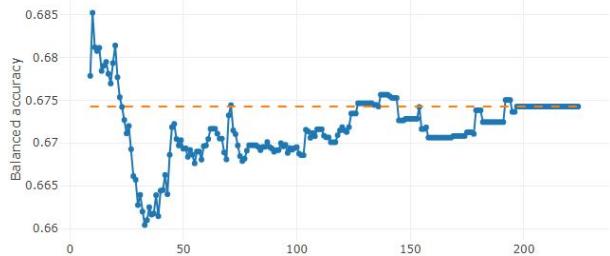


Figure 11: Balanced accuracy for G^t , $9 \leq \alpha \leq 224$. The dotted line shows the balanced accuracy for G .

5.6 Improvement of Graph Construction

In Figure 6 we can see that G not only contains vertices of degree 9, but also of degree 20 times as large. After trimming the edges, the graph is close to a 9- or 10-regular graph (see Figure 7). In particular, the maximum degree is 22, a little more than twice the most frequent degree arising in G^t .

Figure 10 shows the distribution of the weighted degree in G^n , the normalized version of G . Again, the maximum degree is a little more than twice the most frequent degree arising, whereas the minimum degree is comparatively small, i.e. the graph is not close to a regular weighted graph. However, the shape of the distribution is quite similar to the shape seen in Figure 7.

Figure 11 shows the balanced accuracy for G^t

where α is ranging from 9, the minimum degree in G , to 224, the maximum degree. For small α , G^t is close to a regular graph, i.e. hubs were successfully eliminated yielding a good result. Furthermore, for large α , G^t is very similar to G and hence the result is approximately the same. However, there is a notable global minimum around $\alpha = 35$, suggesting that hub removal should be done either completely or not at all.

5.7 Towards the Fully Connected Graph

Due to the high amount of memory needed, we cannot construct the fully connected weighted graph proposed by Zhu and Ghahramani (2002), that is the graph given by taking the similarity matrix W along with the similarity function σ_γ as adjacency

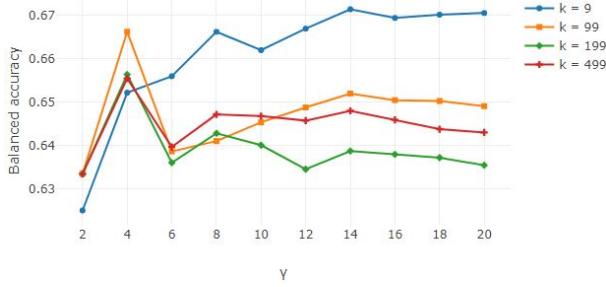


Figure 12: Bal. Accuracy for G_γ , $1 \leq \frac{\gamma}{2} \leq 10$, and multiple values for k .

matrix A . However, the weighted k NN graph G_γ with large k is a good approximation as $\sigma_\gamma(x_i, x_j)$ is strictly decreasing in $\|x_i - x_j\|_2$ and hence, only the edges having small weight are missing.

As an example, Figure 12 shows the balanced accuracy for $k \in \{9, 99, 199, 499\}$ ($k = 499$ is very close to the maximum value possible on our hardware). We can see that large k harms the classification, thus confirming the results on sparse adjacency matrices mentioned in section 3.2.

We do not rule out the fact that there could be a state change as $k \approx n$ where the information flow improves drastically and causes the SSL classification performance to spike. We leave this as an open question for future work.

6 Conclusion

In this paper, we study label propagation for sentiment detection on word vectors obtained by training a FastText model as well as by using pre-trained models, which clearly perform worse. We showed empirically that the unweighted 9NN graph performs better on the given task than its weighted counterpart and the approximation of the fully connected weighted graph.

Furthermore, we propose improvements to state-of-the-art methods for the construction of the underlying graph, and show that properly chosen anti-hub routines and mild ϵ -sparsification improves the result. In particular, edge trimming is a fast algorithm to transform a k NN graph into a more regular one.

7 Future Work

Possible directions for future research include the development of an online label propagation algorithm based on entropy and data quantization (in the spirit of (Valko et al., 2012)). The goal is to

improve classification performance for situations where the word vector embedding of the given data does not fulfill the cluster assumption perfectly. Furthermore, the ability of being able to deal with streaming data is a highly attractive add-on for practical applications of SSL models.

Another interesting idea is the search for metrics quantifying the cluster assumption for the embedded data, as discussed above. This can be supplemented by an analysis of the performance of label propagation conditioned on the scores provided by the metrics found above and hence, by the relevance of the word embedding.

Datasets which can be used to examine the performance of the given SSL algorithm include the annotations on polarity shifters by (Schulder et al., 2018) and the domain-specific corpora for computational social science by (Hamilton et al., 2016).

References

- [Bengio et al.2006] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2006. 11 label propagation and quadratic criterion.
- [Beygelzimer et al.2006] Alina Beygelzimer, Sham Kakade, and John Langford. 2006. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104. ACM.
- [Bojanowski et al.2016] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.
- [Chapelle et al.2009] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.
- [Chen et al.2009] Jie Chen, Haw-ren Fang, and Yousef Saad. 2009. Fast approximate knn graph construction for high dimensional data via recursive lanczos bisection. *Journal of Machine Learning Research*, 10(Sep):1989–2012.
- [de Sousa et al.2013] Celso André R de Sousa, Solange O Rezende, and Gustavo EAPA Batista. 2013. Influence of graph construction on semi-supervised learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 160–175. Springer.
- [dos Santos and Gatti2014] Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

- [Dua and Graff2017] Dheeru Dua and Casey Graff. 2017. UCI machine learning repository.
- [Fredman and Tarjan1987] Michael L Fredman and Robert Endre Tarjan. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615.
- [Gamon2004] Michael Gamon. 2004. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of the 20th international conference on Computational Linguistics*, page 841. Association for Computational Linguistics.
- [Giulianelli2017] Mario Giulianelli. 2017. Semi-supervised emotion lexicon expansion with label propagation and specialized word embeddings. *CoRR*, abs/1708.03910.
- [Goldberg and Zhu2006] Andrew B Goldberg and Xiaojin Zhu. 2006. Seeing stars when there aren’t many stars: graph-based semi-supervised learning for sentiment categorization. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, pages 45–52. Association for Computational Linguistics.
- [Hamilton et al.2016] William L Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 595. NIH Public Access.
- [Huang and Jebara2007] Bert Huang and Tony Jebara. 2007. Loopy belief propagation for bipartite maximum weight b-matching. In *Artificial Intelligence and Statistics*, pages 195–202.
- [Jebara et al.2009] Tony Jebara, Jun Wang, and Shih-Fu Chang. 2009. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 441–448. ACM.
- [Joulin et al.2016] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.
- [Klenner et al.2009] Manfred Klenner, Angela Fahrni, and Stefanos Petrakis. 2009. Polart: A robust tool for sentiment analysis. In *Proceedings of the 17th Nordic Conference of Computational Linguistics (NODALIDA 2009)*, pages 235–238.
- [Maaten and Hinton2008] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.
- [Matsumoto et al.2005] Shotaro Matsumoto, Hiroya Takamura, and Manabu Okumura. 2005. Sentiment classification using word sub-sequences and dependency sub-trees. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 301–311. Springer.
- [Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- [Ozaki et al.2011] Kohei Ozaki, Masashi Shimbo, Mamoru Komachi, and Yuji Matsumoto. 2011. Using the mutual k-nearest neighbor graphs for semi-supervised classification on natural language data. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 154–162. Association for Computational Linguistics.
- [Pang et al.2002] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.
- [Pearson1901] Karl Pearson. 1901. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.
- [Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- [Peters et al.2018] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.
- [Radovanović et al.2010] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11(Sep):2487–2531.
- [Ram et al.2009] Parikshit Ram, Dongryeol Lee, William March, and Alexander G Gray. 2009. Linear-time algorithms for pairwise statistical problems. In *Advances in Neural Information Processing Systems*, pages 1527–1535.
- [Rao and Ravichandran2009] Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 675–682. Association for Computational Linguistics.

- [Ravi and Diao2016] Sujith Ravi and Qiming Diao. 2016. Large scale distributed semi-supervised learning using streaming approximation. In *Artificial Intelligence and Statistics*, pages 519–528.
- [Remus et al.2010] Robert Remus, Uwe Quasthoff, and Gerhard Heyer. 2010. Sentiws-a publicly available german-language resource for sentiment analysis. In *LREC*. Citeseer.
- [Ren et al.2012] Yong Ren, Nobuhiro Kaji, Naoki Yoshinaga, Masashi TOYODA, and Masaru KITSUREGAWA. 2012. Semi-supervised sentiment classification in resource-scarce language: A comparative study (. . DE, , 112(172):59–64.
- [Schulder et al.2018] Marc Schulder, Michael Wiegand, Josef Ruppenhofer, and Stephanie Köser. 2018. Introducing a lexicon of verbal polarity shifters for english.
- [Tabei et al.2010] Yasuo Tabei, Takeaki Uno, Masashi Sugiyama, and Koji Tsuda. 2010. Single versus multiple sorting in all pairs similarity search. In Masashi Sugiyama and Qiang Yang, editors, *Proceedings of 2nd Asian Conference on Machine Learning*, volume 13 of *Proceedings of Machine Learning Research*, pages 145–160, Tokyo, Japan, 08–10 Nov. PMLR.
- [Taboada et al.2011] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.
- [Valko et al.2012] Michal Valko, Branislav Kveton, Ling Huang, and Daniel Ting. 2012. Online semi-supervised learning on quantized graphs. *arXiv preprint arXiv:1203.3522*.
- [Van Der Maaten2014] Laurens Van Der Maaten. 2014. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245.
- [Vega-Oliveros et al.2014] Didier A Vega-Oliveros, Lilian Berton, Andre Mantini Eberle, Alneu de Andrade Lopes, and Liang Zhao. 2014. Regular graph construction for semi-supervised learning. In *Journal of physics: Conference series*, volume 490, page 012022. IOP Publishing.
- [Waltinger2010a] Ulli Waltinger. 2010a. Germanpolarityclues: A lexical resource for german sentiment analysis. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta, May. electronic proceedings.
- [Waltinger2010b] Ulli Waltinger. 2010b. Sentiment analysis reloaded: A comparative study on sentiment polarity identification combining machine learning and subjectivity features. In *Proceedings of the 6th International Conference on Web Information Systems and Technologies (WEBIST '10)*, Valencia, Spain, April.
- [Zhu and Ghahramani2002] Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation.
- [Zhu2008] X Zhu. 2008. Semi-supervised learning literature survey, department of computer sciences, university of wisconsin. Technical report, Madison, 2008 (Technical Report, 1530).

Detecting the boundaries of sentence-like units on spoken German

Josef Ruppenhofer

Institut für Deutsche Sprache

R5, 6-13

D-68161 Mannheim

ruppenhofer@ids-mannheim.de

Ines Rehbein

Institut für Deutsche Sprache

R5, 6-13

D-68161 Mannheim

rehbein@ids-mannheim.de

Abstract

Automatic division of spoken language transcripts into sentence-like units is a challenging problem, caused by disfluencies, ungrammatical structures and the lack of punctuation. We present experiments on dividing up German spoken dialogues where we investigate the impact of task setup and data representation, encoding of context information as well as different model architectures for this task.

1 Introduction

Being able to structure natural spoken discourse into sentence-like units (SLUs) is desirable not only from a theoretical point of view, but is also a key requirement for enabling research in corpus linguistics as well as the application of Natural Language Processing tools (e.g. POS-tagging and parsing) to transcripts of spoken language. While various proposals have been made for how to divide spoken language in corpora into smaller units, typically these divisions were not guided by syntactic considerations. Instead, division into inter-pausal units is common (e.g. Hamaker et al. (1998) for the Switchboard corpus). Until recently, for most languages no well-established system existed for detecting boundaries between sentence-like units that is both theoretically well-founded and practically operationalizable for large and diverse corpora of spoken interaction.

For German, the SegCor project (Westpfahl and Gorisch, 2018; Westpfahl et al., 2019) developed guidelines for dividing transcribed speech into sentence-like units using the topological field model of German surface syntax. Schmidt and Westpfahl (2018) subsequently presented a corpus-based study on how well the length of gaps between utterances can predict the syntactic boundaries annotated in the SegCor corpus.

In this work, we take up the challenge of automatically detecting boundaries between SLUs on the spoken German of the SegCor transcripts. Further, we apply our system not only to the question whether a gap, a long silence, coincides with a syntactic boundary but to all boundaries in general, including the ones that occur in continuous speech, such as interruptions and aborted utterances.

This paper proceeds as follows. We discuss related work in section 2 and present our dataset in section 3. In sections 4 and 5 we discuss the task formulations we employ and the features we use. Our experiments and their results are described in section 6, followed by a conclusion in section 7.

2 Related Work

In the realm of medially written language, the most closely related task is sentence boundary detection (SBD). Typically, this has been framed as deciding for a closed class of punctuation symbols (mainly '.', '?', '!') whether they represent the end of a sentence or not, with abbreviations constituting one of the key sources of error. While traditionally very high accuracies were reported, Read et al. (2012) show in their overview of SBD that performance can be significantly worse on text other than news, with machine learning-based systems often being less robust than rule-based or hybrid systems. Comparing Wikipedia pages to topically related blogs, they also show that within the same domain, sentence-boundary detection performs less well the more informal the text type is. Read et al. (2012) observe that the traditional framing of the problem overlooks all the cases where sentences or rather sentence-like units, text sentences in the sense of Nunberg (1990), end without a punctuation symbol: on the ‘standard’ texts in their collection, this affects 12.3% of sentences. Read et al. (2012) therefore argue for a more general approach ‘which considers the positions after every character as a potential boundary point’.

In the domain of medially spoken language, the detection of sentence-like units may use both textual and prosodic features. Gotoh and Renals (2000) performed experiments with HMMs on reference transcripts from BBC radio and tv programs which included repeated and incorrect speech as well as disfluencies. They also constructed an alternative pause duration model alone based on speech recogniser output aligned with the transcripts. The pause duration model outperformed the language modelling approach, while a combination of the two models provided further performance gains. Precision and recall scores of over 70% were attained for the task of deciding for each word whether it represents the last word of a sentence. In his work on sentence boundary detection on Czech radio news and discussion programs, Kolář (2008) similarly finds that combining several models works best.

Liu et al. (2005) evaluate the performance of a CRF-model on two English corpora (conversational telephone speech and broadcast news speech) on both human transcriptions and automatic speech recognition output. Their experiments show that the use of prosody improves performance over the use of word n-grams alone and that the addition of further features e.g. on pos-tags provides another improvement.

Roark et al. (2006) use a re-ranking approach to the detection of SLU boundaries. In a two stage approach, they first fix a subset of the word boundaries as points of division, yielding subsequences between fixed points, which they call fields. In the second stage, candidate boundaries within the fields are generated and then ranked.

In our own experiments, we will experiment with various features and task parameters used by prior work such as e.g. POS, gap/pause-length, use of left and/or right context etc. In addition, we also explore extra features available with our dataset.

3 Dataset

The data used here is unlike most of the material used in related work in that it represents conversational speech that was furthermore recorded in non-laboratory settings. Also, it is characterized by interactions between two or more speakers. Since tools based on the automatic processing of the audio signal do not work all that well on our data, we instead work with the transcripts only. Our dataset consists of 33 documents with more than 54,000

lexical tokens originating from the FOLK corpus (Schmidt, 2014) that were divided into sentence-like units by the SegCor project. This data set was doubly annotated and disagreements were adjudicated (Westpfahl and Gorisch, 2018). Note that to avoid confusion, we reserve the term "segment" and related forms for the division of speech into chunks by the transcribers that was guided by silences in the speech signal. For the division of the material into sentence-like units we will use the term "SLU boundary detection".

The raw FOLK transcripts, which we take as our input and which lack SLU-boundaries, follow the cGAT conventions (Schmidt et al., 2015). Accordingly, the data uses "contributions" and "segments" as the fundamental units in the data structure. Segments of speech are the original units of transcription: transcribers are instructed to select them as chunks that can be transcribed in one go given cognitive load and useability of the transcription environment. Crucially, segment boundaries should be placed at word boundaries or at the beginning or end of pauses. Like segments, contributions are defined without any reference to syntactic considerations (Schmidt et al., 2015, 8):

'A contribution in a cGAT transcript comprises all immediately consecutive segments attributed to a speaker. Contributions should not be confused with sentences, which are units of written language. Instead, they are to be understood as dialogue contributions.'

Pauses (silences up to 0.2s) may occur between separate contributions but also within a contribution. Gaps, silences longer than 0.2s, always separate contributions in cGAT.

The relation between the input representation in terms of contributions and the intended output representation in terms of sentence-like units is not always one to one. Common deviations are as follows. First, a contribution may correspond to several SLUs as illustrated by (1).

- (1) 1 contribution : n SLUs
 - a. < c >h ich weiß net ich glaub eher nich h h< /c >
 - b. < s >h ich weiß net< /s >
 < s > ich glaub eher nich h h< /s >
 - c. 'I don't know. I rather think not.'

Second, several contributions may jointly correspond to one SLU.

(2) *n* contributions : 1 SLU

- a. < c >der beschäftigt sich< /c >
< c >(0.85) < /c >
< c >zwei minuten mit dem< /c >
- b. < s > der beschäftigt sich (0.85)
zwei minuten mit dem < /s >
- c. ‘He occupies himself with that one
for two minutes.’

Both situations may also occur in combination so that we get *n* : *m*-relations between contributions and SLUs.

To decide on SLU boundaries, we can use not only the transcribed word forms but also some further kinds of information about the tokens, which we will use as features (cf. section 5). Further, while we do not use acoustic features such as word durations and pitch contours, the transcript does give us access to temporal information that has proved useful in previous work (Gotoh and Renals, 2000). We encode pause length and, since we know which tokens are produced by which speaker, we also introduce turn boundaries into our representation.

4 Task formulations

We can approach the SLU boundary detection problem in various different ways. We discuss the major points of variation in what follows.

4.1 Granularity

In one line of experiments (coarse), we predict only whether a token is followed by some type of syntactic boundary (B) or not (O). In another line (fine), we also distinguish between several types of boundaries. From Westpfahl and Gorisch (2018), we adopt the following B(oundary) types.

S Simple sentential units consist of exactly one clause. In terms of word order, the clause may be of any of the types V1 (verb initial), V2 (verb second), V1/2 (cases that are unclear between V1 and V2) or in rare cases VL (verb last). The clauses may not have any dependent clauses.

C Complex sentential units consist of several clauses that are dependent on one another:

Main clauses with subordinate clauses or relative clauses, conditional sentences, reported speech, and matrix-clause with sentient-verbs, complex pre-pre-fields with main clause, discontinuous sentences, and coordinated sentences if and only if the second sentence shows subject or verb ellipsis.

N Non-sentential units are all units that are not structured by a finite verb.

A An utterance which is disrupted, i.e. it opens a projection that subsequently goes unfilled.

U Tokens at the end of a unit whose status could not be categorized as one of the previous four cases.

Since in the context of sequence labeling we need to have a label on every token, we add several further categories of non-boundary labels. In the binary setting, these categories are merged into the non-boundary class (O).

O Words spoken by one of the speakers that are not followed by a boundary.

X is used for different types of non-verbal information: a) speaker turns, and b) pauses. We distinguish between pauses shorter than 0.2 sec and longer pauses. According to cGat, longer pauses always occur between two adjacent contributions and are not assigned to any speaker while shorter pauses are considered to be part of one speaker’s contribution. For instance, the pause in (i) is part of speaker RD’s contribution as they are just pausing speech for the purposes of word finding. By contrast, the pause in (ii) is not assigned to either speaker: it is clear that speaker RD has finished their turn, but speaker LH has not yet taken the floor.

i RD: ich könnte es ja darüber lösen dass
ich das nicht auf das <pause> ko auf die
konten der seefahrer buch sondern auf
ein verrechnungskonto

‘Well, I could fix it in this way that I don’t book it on the acc on the accounts of the sailor but instead to a clearing account’

ii RD: ich versthe nichts davon
‘I don’t know anything about it’
<pause>
LH: okay ...

In our experiments, both pause types are assigned the tag “X”.

4.2 Views

Since our data comes from multi-party conversation it lends itself to two views. On the one hand, we can think of it as an integrated **conversation**, where contributions of speakers alternate, with occasional overlaps. The intuition behind adopting this view on the data is that a speaker’s productions do depend on / respond to what the other speaker says. For instance, responses to questions are often not complete sentential units whether simple or complex but rather consist of non-sentential material. For that reason, it seems important to take into account what interlocutors are saying.

A second, complementary view of the data treats it as a set of **tracks** of speech, each by one specific person. The intuition behind this view is that the sentence-like units are local only to the given speaker’s utterances. For instance, whether a sentence is simple or complex depends only on what the current speaker produces. In adopting a track view (track), we completely ignore the other speaker’s productions.

Both views potentially have problems handling certain kinds of so-called split utterances (Purver et al., 2009). On the conversation view, utterances that are distributed across multiple contributions of the same speaker may be interrupted by contributions of other speakers. On the track view, utterances that are distributed across speakers (that is, co-constructed turns begun by one speaker but finished by another) cannot be recovered.

4.3 Instance creation

We define instances for the classifier either in terms of word **windows** of varying size or in terms of N **merged** contributions.¹

4.4 Model type

As demonstrated by the related work, one established way to approach the SLU boundary detection problem is in terms of **sequence labeling**. The task consists in algorithmically assigning a categorical label to each item in a sequence of observed values. In our task, a token is labeled either as being followed by a boundary or not.

As a baseline approach, we adopt a classical Conditional Random Fields (Lafferty et al., 2001) tagger, using the CRFsuite implementation by

¹Other variations are possible such as creating overlapping instances. For instance, with word windows we could create one instance from words 1-10 and the next from words 2-11 etc. We could proceed similarly in the case of contributions.

Okazaki (2007), for which we provide our own feature engineering.

We compare this system with two more recent neural architectures. The first system is an implementation of the model of Lample et al. (2016), using biLSTMs for input encoding, based on word and character-based embeddings, followed by a CRF layer on top (Reimers and Gurevych, 2017).² The second model, the flair sequence tagger (Akbik et al., 2019), has a similar architecture that also combines biLSTMs and a CRF layer on top. In addition, flair uses *contextual string embeddings* (Akbik et al., 2018) which model words as contextualized sequences of characters, resulting in different embeddings for the same string, depending on its surrounding context.

5 Features

The data encodes the following information that we can use as features in our experiments.

Tokens The simplest feature are the raw transcribed tokens.

POS The SegCor data includes automatically predicted POS tags.

Normalization The normalization layer contains the canonicalized form for the raw tokens. For instance, when an instance of the first person present form of the verb *verstehen* ‘understand’ is pronounced as two syllables, without its final weak syllable, it is transcribed as *ver-steh*. The normalization of the token will be the expected canonical form *verstehe*. Also while all noun tokens appear lowercased in the transcription, they are written with initial capitals on the normalization layer.

Lemma The lemma forms for the transcribed data.

6 Experiments

At the highest level, we divide our experiments depending on the granularity, coarse or fine. Within these high-level groups, we discuss the experiments in sets that address a common research question.

We use 70, 10 and 20% of the data for training, development and testing, respectively. We do not split up individual transcriptions but put them whole into either train, dev or test. This makes the task slightly harder as we test on data from new speakers that have not been seen during training, and on new topics that are not included in

²<https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf/>

ID	View	Instances	Macro Acc	Macro F1	F1 B	F1 O	Description
1	track	single	83.75	45.58	0.00	91.16	majority class, i.e. no boundaries
2	track	single	89.98	74.99	55.63	94.35	boundary at end of contribution

Table 1: Results for rule-based baselines (coarse-grained, track: track-view; singe: single contributions)

	ID	View	Instances	Macro Acc	Macro F1	F1 B	F1 O	context	features
instance creation	3	track	single	94.20	87.25	77.84	96.67	+/-2	word,pos
	4	track	single	93.66	86.04	75.73	96.36	+/-1	word,pos
	5	track	merged	94.69	88.33	79.71	96.95	+/-2	word,pos
	6	track	merged	93.99	86.74	76.93	96.54	+/-1	word,pos
	7	track	window	94.01	86.58	76.59	96.57	+/-2	word,pos
context size	8	conv.	window	93.54	85.42	74.54	96.30	+/-2	word,pos
	9	track	merged	94.78	88.56	80.13	97.00	+2	word,pos
	10	track	merged	93.53	85.60	74.90	96.29	+1	word,pos
	11	track	merged	89.21	73.25	52.58	93.91	-1	word,pos
single feats.	12	track	merged	88.75	72.86	52.09	93.63	-2	word,pos
	13	track	merged	93.86	85.87	75.25	96.50	+/-2	word
	14	track	merged	93.86	86.46	76.46	96.47	+/-2	pos
	15	track	merged	93.76	85.89	75.36	96.43	+/-2	lemma
turn norm.	16	track	merged	94.16	86.88	77.10	96.66	+/-2	normalization
	17	track	single	94.14	87.15	77.68	96.63	+/-2	norm, pos
	18	track	merged	94.78	88.52	80.05	97.00	+/-2	norm, pos
	19	track	merged	92.56	84.38	73.07	95.68	+/-2	word, pos; no turns

Table 2: Results for sequence labeling with CRFsuite (coarse-grained, track-view; conv.: conversation; merged: 5 merged contributions; window: 10-word windows)

ID	View	Instances	Macro Acc	Macro F1	F1 B	F1 O	Embeddings	Schema
20	track	merged	94.14	87.06	77.48	96.63	Reimers2017	word
21	track	merged	94.36	87.69	78.63	96.75	Reimers2017	norm

Table 3: Results for biLSTM-CRF sequence tagger (Lample et al., 2016) (coarse-grained, track-view)

ID	View	Instances	Macro Acc	Macro F1	F1 B	F1 O	Embeddings
22	track	merged5	95.07	89.59	82.05	97.14	fasttext+flair
23	track	merged5	92.28	83.42	71.30	95.54	fasttext
24	track	merged5	94.83	89.28	81.56	97.00	fasttext+custom
25	track	merged5	95.43	90.23	83.11	97.36	fasttext+flair+custom

Table 4: Results for flair’s sequence tagger with contextual string embeddings (coarse-grained, track-view)

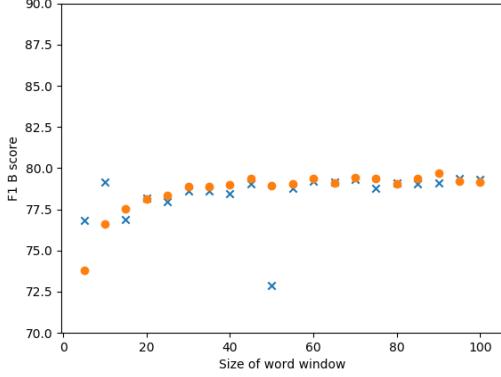


Figure 1: F1 B-score for word windows of various sizes (dots: conversations; x’s: tracks; step size=5; CRFsuite)

the training set. Thus, the classifier cannot adapt to speaker-specific features and might encounter a larger amount of unknown words. However, this setting is more realistic and will give us a better estimate of what to expect when applying our models to new data.

For all non-deterministic models, we report results averaged over three runs for each configuration.

6.1 Coarse-grained classification

Baselines In addition to using CRFsuite as a baseline, we calculated the following two rule-based baselines (table 1). Baseline 1 always assigns the majority class (no boundary) while baseline 2 predicts a boundary at the last token in each contribution. Recall that the contributions are not *gold* sentences but can also cross syntactic boundaries, which is shown by the less-than-perfect results for baseline 2 (89.98% acc. and 55.63% F1 for the Boundary class). As will be shown by the experiments to follow, machine-learning based systems, unsurprisingly, can yield much better results.

Views and instance creation First, we investigate the impact of view and instance creation on the performance for varying window sizes. Figure 1 plots the F1 scores for Boundaries relative to growing sizes of word windows used to construct instances. The results are very similar regardless of whether we use the conversational view or the track view.

Figure 2 shows the development of the F1 B-score in relation to the number of contributions that are assembled into one instance. We observe that,

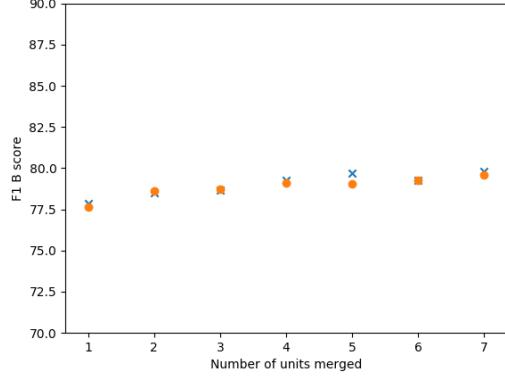


Figure 2: F1 B-score for track view in relation to contributions merged (CRFsuite) (dots: conversations; x’s: tracks)

here too, the results hardly differ between the track view and the conversational view.

While it should not matter much in practice, we choose to mainly work with the combination of merging segments on the track view for the remainder of the paper since the highest F1-score that we obtained in these experiments come from this combination.

Importance of Context We now focus on the question where in the context the relevant information for boundary detection is. Thus, the second block of experiments varies the context relative to our reference experiments 5 and 6 (Table 2), using either only the left or the right context, or no context at all. The contrast between the results for the experiments with one-sided context shows that the right context is clearly more important than the left one and that the left context by itself does not hold very much information to begin with.

Individual features Experiments 13–16 present results for runs with individual features. The results show that not all forms of generalizing over the concrete tokens work equally well. The automatically assigned lemmatization probably is worst because on our data it is also often wrong. POS-tags are better but the normalized text representation, though also automatically assigned, is best.

Normalization Following on the observation about the utility of normalization, in experiments 17 and 18, we use the normalization layer instead of the transcribed tokens in combination with POS tags. When contrasting the results of these experiments with those of exp. 3 and 5, we see that

normalization gives slightly better results only in the second setting. Given that normalization is also time-consuming, in later experiments we will not use the normalization layer but instead use the transcribed speech as input.

Importance of Sequencing Information In experiment 19, we use a version of the data from which, unlike for all other track view-based experiments, the representation of turns has been eliminated. Compared to the matched basic experiment 5, we see a significant drop in Macro F1 and the F1 for the B(oundary) class, which underscores the importance of including information on turns.

Classical CRF vs. biLSTM-CRF Recent advances in NLP have shown the expressive power of neural networks. We thus compare the performance of the classical CRF sequence tagger to two neural systems, the one of (Lample et al., 2016; Reimers and Gurevych, 2017) and the flair sequence tagger, as described in Section 4.4.

Table 3 shows that the neural biLSTM-CRF does not always improve results over the classical CRF. The first system uses word and character-based embeddings as features and predicts the binary labels $\{B, O\}$. This configuration does not outperform CRFsuite configurations such as 5 where we also use POS tags as features, in addition to the word tokens.

The biLSTM-CRF can make better use of the normalization, as shown in experiment 21. Compared to experiment 16, we gain 1.5% in performance. Both systems, however, are outperformed by the flair sequence tagger with contextual string embeddings (Table 4, exp 22).

Embeddings used Given that flair outperforms the model of Lample et al. (2016) despite their similar architecture, we now explore variation around the embeddings used in flair. Experiment 23 shows the value of flair’s contextual string embeddings: without them performance decreases by more than 10% for F1 B (see exp. 22).

In our next experiment, we want to test whether we can increase performance by training our own contextual string embeddings on text that is more similar to our data. For this, we train flair embeddings for 20 epochs on ca. 11 million ‘sentences’ extracted from the open subtitles corpus (Lison and Tiedemann, 2016) and an in-house twitter dataset. These sentences were filtered to be at most 60 char-

acters long and to contain no more than one comma and one period, question mark or exclamation mark. The punctuation marks were removed before training and the data was lowercased. In experiment 24 we use these custom embeddings in combination with fasttext only without the default forward and backward embeddings provided by the flair library.

The results show that the custom embeddings are quite good on their own (exp. 24). Combining them with flair’s pretrained embeddings further improves results, showing that our custom embeddings contain complementary information (exp. 25). While the results suggest that the use of more domain-similar contextual string embeddings is beneficial, we cannot be sure that the improvements are really due to domain similarity. To test this in future work, we will need to compare our results to another type of custom embeddings trained on a corpus of equal size but with different properties that are less similar to spoken language, such as newspaper text.

6.2 Fine-grained classification

We now turn to the fine-grained setting which distinguishes between five kinds of boundary labels. For ease of presentation and since the non-boundary labels are not important to us, we will report F1 scores for each boundary label with the exception of the U(ninterpretable) class, which is conceptually ill-defined since by definition it is unclear whether, and what kind of, a boundary occurs. As well as the global Macro F1 and Macro Accuracy scores, we also report a score “Macro F1 B” which constitutes the macro average over the boundary labels, including U.

As a reference for the flair sequence tagger, Table 5 shows results for CRFsuite for the trackwise view and instances formed by merging contributions.³ As shown by the difference in F1-scores between the fine-grained and the coarse-grained settings from Table 2, the fine-grained task is much harder. Again, using word windows of size 10 for instance creation is worse than merging contributions.

The gap between CRFSuite and the neural system shows the potential of the contextual string embeddings: Flair outperforms CRFSuite substantially (cf. exp. 29 vs. 27). Focusing on the flair results, we see that the performance on the individual boundary types strongly depends on their fre-

³For lack of space we do not report results for the biLSTM-CRF model of (Lample et al., 2016; Reimers and Gurevych, 2017) which again was outperformed by flair.

Id	View	Instances	Macro F1	Macro Acc	F1 A	F1 C	F1 N	F1 S	Macro F1 B
26	track	window	58.51	97.61	22.79	26.32	73.55	51.01	43.42
27	track	merged	58.15	97.65	25.30	26.24	73.92	52.20	44.20

Table 5: Results for fine-grained sequence labeling with CRFsuite

Id	View	Instances	Macro F1	Macro Acc	F1 A	F1 C	F1 N	F1 S	Macro F1 B
28	track	window	68.59	98.10	42.82	45.76	80.16	66.34	56.69
29	track	merged5	70.24	98.22	42.93	50.49	81.59	68.95	58.98

Table 6: Results for fine-grained sequence labeling with flair

quency: results for the rarer classes A(borted) and C(omplex) are substantially lower than the ones for the more frequent classes N(on-sentential) and S(imple).

6.3 Error analysis

To get a sense of what the flair sequence tagger is able to learn, in Table 7 we take a look at the confusion matrix for the best fine-grained experiment 29. Among the boundary classes, A(borted) segments are mostly not recognized as having any kind of boundary, i.e. they receive the label O; smaller subsets of true A's are mistaken for non-sentential units or simple sentences. When A's get confused for O's, this often seems to be due to the boundary token being an incomplete, partial word such as *a* or *we*.

For C(omplex) segments, being mistaken for a simple sentence (S) is the most common error, before not being recognized as any kind of bounded segment. One class of C-S confusions arises when subordinate complement clauses lack a complementizer and verb-second word order is used, as in example (3).

- (3) < c > ich wiederhole das sind tonsteine
 (.) mit eingelagerten kalksandsteinbänke

	A	C	N	O	S	U	X	Total
A	57	3	12	93	20	0	0	185
C	0	98	5	61	75	1	0	150
N	5	6	584	78	36	0	0	709
O	12	26	102	8836	84	2	0	9062
S	7	24	17	128	439	0	0	615
U	1	0	4	6	2	9	0	22
X	0	0	0	0	0	2105	2105	

Table 7: Confusion matrix for best fine-grained run (exp. 29; across: predicted; down: gold)

< /c >

'I repeat [that] these are mudstones with embedded banks of sand-lime brick.'

Finally, for S(imple) sentences not being recognized as a bounded segment is the most common error. One subtype of this error that we recognize are cases where the final token is an unlikely one. Consider example 4, whose true labeling is given. The error that flair makes is to include all the tokens in a single S(imple) sentence, even though this means that the resulting simple sentence incorrectly has two finite verbs. Potentially, the error occurs because the adverb *angeblich* 'supposedly' is an unlikely sentence ending token. In example 5, the initial complex sentence is correctly recognized but the following simple sentence receives no boundary label even though it is followed by a change of turn. Again, the problem seems to be that the subject pronoun *er* 'he' is an unlikely sentence-final token. Other instances concern elliptical cases where modal verbs occur sentence-finally without an infinitival complement (e.g. *die müssen* 'They must'). A second subtype of error consists of infrequent sentence types. Consider the example in 6. This is an unusual case because it is a free-standing subordinate clause, which gets treated as a simple sentence according to the SegCor guidelines. Flair marks no boundary here, which results in the main clause of the following complex sentence having two finite verbs.

- (4) < s >da war des doch fast die ältesten
mutter angeblich< /s >< s >mit sieben-
nugsechzig hat se s kind gekriegt oder
so< /s >

'She was almost the oldest mother there supposedly. She had the child at sixty-seven or thereabouts.'

- (5) < c > was ich gelesen hab(.) muss immer
derjenige äh zu lebzeiten schon seine ein-
verständnis abgegeben< c / > < s > nur die
nimmt er< / s >
'From what I have read that person al-
ways has to give their consent during their
lifeime. Only those ones he accepts.'
- (6) < s > ob ich des hinkriech < / s >
'[I am wondering] if I can manage that.'

Finally, we want to note that sentence boundary labeling cannot be done perfectly by humans and that its difficulty is variable across text types. Westpfahl and Gorisch (2018) report an average kappa of 0.69 across 8 transcripts. Across the transcripts, the kappa value ranges from 0.53 for a conflictual interaction to 0.76 for a reading child. While Westpfahl and Gorisch (2018) give no breakdown of which confusions among boundary types are most frequent for their human annotators, they do show a further complication of the task: the different sentence types are distributed differently across different text types and their specific properties also vary by text type. For instance, in so-called expert talk, simple sentences are longer than in other texts. Taken together, these considerations underline the challenge in the task we tackle.

7 Conclusions and Future Work

We have investigated the problem of detecting SLUs in spoken German. We found that the choice of data representation for the classifier is important: small word windows perform worse than larger ones but the merging of contributions performs well in a robust way, no matter the size. Further, we found that the main challenge of the task is to recognize sentence beginnings: the right context is much more important than the left context. We also verified that using information on turns is important. Finally, we found that augmenting flair's embeddings with domain-similar custom embeddings further enhances performance.

Given the success of the contextual string embeddings, in future work we would like to investigate whether other contextualized representations such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) can yield further improvements.

Another approach to SLU boundary detection frames it in terms of **sequence-to-sequence** learning, using attention-based neural encoder-decoder models (Bahdanau et al., 2015). Here, a model is

trained to convert sequences from one domain to sequences in another domain. A typical application scenario for this class of models is machine translation. In our case, we would translate spoken German utterances lacking SLU boundaries into speech with SLU boundaries. While initial experiments showed that sequence-to-sequence models are also able to learn boundaries for spoken utterances, we did not have enough training data to achieve competitive results. We will pursue this avenue in future work, using additional naturalistic as well as synthetically created training data.

Acknowledgments

This research has been partially supported by the Leibniz Science Campus "Empirical Linguistics and Computational Modeling", funded by the Leibniz Association under grant no. SAS-2015-IDS-LWC and by the Ministry of Science, Research, and Art (MWK) of the state of Baden-Württemberg.

References

- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.
- Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota, June. Association for Computational Linguistics.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, ICLR 2015.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *The 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, pages 4171–4186.
- Yoshihiko Gotoh and Steve Renals. 2000. Sentence boundary detection in broadcast speech transcripts. In *in Proc. of ISCA Workshop: Automatic Speech Recognition: Challenges for the new Millennium ASR-2000*, pages 228–235.

- Jonathan Hamaker, Yu Zeng, and Joseph Picone. 1998. Rules and guidelines for transcription and segmentation of the switchboard large vocabulary conversational speech recognition corpus. Technical report.
- Jáchym Kolář. 2008. *Automatic Segmentation of Speech into Sentence-like Units*. Ph.D. thesis, PhD Thesis, University of West Bohemia, Pilsen, Czech Republic.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.
- Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).
- Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. 2005. Using conditional random fields for sentence boundary detection in speech. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 451–458. Association for Computational Linguistics.
- Geoffrey Nunberg. 1990. *The Linguistics of Punctuation*. CSLI, 01.
- Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *The 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT 2018, pages 2227–2237.
- Matthew Purver, Christine Howes, Patrick G. T. Healey, and Eleni Gregoromichelaki. 2009. Split utterances in dialogue: A corpus study. In *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '09, pages 262–271, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Jonathon Read, Rebecca Dridan, Stephan Oepen, and Lars Jørgen Solberg. 2012. Sentence boundary detection: A long solved problem? In *Proceedings of COLING 2012: Posters*, pages 985–994, Mumbai, India, December. The COLING 2012 Organizing Committee.
- Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark, 09.
- Brian Roark, Yang Liu, Mary Harper, Robin Stewart, Matthew Lease, Matthew Snover, Izhak Shafran, Bannie Dorr, John Hale, Anna Krasnyanskaya, and Lisa Yung. 2006. Reranking for sentence boundary detection in conversational speech. In *2006 IEEE International Conference on Acoustics, Speech, and Signal Processing - Proceedings*, volume 1, 12.
- Thomas Schmidt and Swantje Westpfahl. 2018. A study on gaps and syntactic boundaries in spoken interaction. In Adrien Barbaresi, Hanno Biber, Friedrich Neubarth, and Rainer Osswald, editors, *The 14th Conference on Natural Language Processing*, KONVENS 2018, pages 40 – 49. Austrian academy of sciences, Vienna, Austria.
- Thomas Schmidt, Wilfried Schütte, and Jenny Winterscheid. 2015. cgat. konventionen für das computergestützte transkribieren in anlehnung an das gesprächsanalytische transkriptionssystem 2 (gat2). Working paper, IDS Mannheim, Mannheim.
- Thomas Schmidt. 2014. The research and teaching corpus of spoken german – folk. In *The 9th conference on international language resources and evaluation*, LREC 2014, pages 383 – 387, Reykjavik. European Language Resources Association (ELRA).
- Swantje Westpfahl and Jan Gorisch. 2018. A syntax-based scheme for the annotation and segmentation of german spoken language interactions. In *The Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions*, LAW-MWE-CxG 2018, pages 109 – 120. Association for Computational Linguistics, Stroudsburg, PA, USA.
- Swantje Westpfahl, Thomas Schmidt, Anton Borlinghaus, and Hanna Strub. 2019. Guideline: syntaktische segmentierung in folker. Working paper, Leibniz-Institut für Deutsche Sprache (IDS), Mannheim.

Dependency Trees for Greenlandic

Eckhard Bick

University of Southern Denmark
ecckhard.bick@mail.dk

Abstract

This paper presents a descriptive system for dependency structures in Greenlandic and proposes a method for implementing it using Constraint Grammar (CG) rules. Our approach aims at reconciling traditional dependency syntax with the polysynthetic morphology of Greenlandic by introducing a novel, morphologically informed tokenization model. For instance, verb-incorporated nominal arguments and adverbials are treated as clause-level constituents rather than morphemes. We discuss and evaluate our alternative tokenization in a cross-language perspective, arguing that the method allows the construction of more universal dependency trees, facilitating both lexical and syntactic transfer in a machine translation (MT) context.

1 Introduction

As a polysynthetic language, Greenlandic has a very low word/sentence ratio, with personal pronouns, prepositions and subordinating conjunctions largely replaced by inflection, and a rich affixation morphology, where each word root can take many bound affix morphemes. Although affixes cannot occur in isolation, they are semantically equivalent to real words in other languages, covering lexical ground otherwise occupied by verbs, nouns, adjectives, adverbs and quantifiers. In addition, a number of enclitic particles, among them the two main coordinators, are also orthographically attached to the

preceding word. As a result, many words have what appears to be internal syntactic structure, joining for instance an incorporated indefinite object with a transitive verb and a modal. In an English translation, such words will end up as noun phrases, verb phrases or even entire clauses or sentences:

Elsip (Else) *Kaali* (Karl)
putumavallaarnasugalugu (since she believes he has had too much to drink)
biileqqunngilaa (forbids him to drive)

Rather than restricting syntactic analysis to word-relations, and postulating a completely separate (morphotactic) grammar for word formation, we therefore advocate splitting Greenlandic words into functional units, with dependency relations and ordering rules holding all the way down to (non-inflexional) morphemes. Thus, for all intents and purposes, we will treat roots and affixes as "words" in the dependency grammar approach presented here¹.

In this approach, we follow Compton & Pittman (2010), who also note syntactic principles, such as ordering rules and positional scope, in Inuit word formation:

"However, the presence of an extra layer of computation in the grammar (i.e., a generative morphological component) raises questions about the role of the syntactic component in such languages. *In particular, it is not clear that the*

¹ In a sense, morpheme chaining within a Greenlandic word is *more* rather than less syntactic than word chaining at the sentence level, given the strict rules governing morpheme ordering and the fact that meaning is order-sensitive.

operations of such a morphological component are in any way different from those of syntax.” (p2)

Rejecting the notion of morphological or syntactic words (p7), they refer to Halle & Marantz (1993) for the concept of “syntax all the way down”, and treat words as Chomskyan “syntactic phrases”, i.e. construction steps rather than absolute units.

In a similar vein, Sadock (1980) advocates pre-affixal syntax², claiming that (Greenlandic) noun-incorporating verbs should not be represented at the deep-structure level (but rather broken up) for syntactic reasons: Incorporated objects (incO's) can occur outside the verb, in the instrumental case (INS), and incO's can be modified by outside modifiers agreeing with the case (INS) and number that the incO would have had in isolation. Also, incorporation of inflected forms is possible, and the type of incorporated argument has syntactic consequences – incorporated objects have their modifier left of the verb, subject complements have it to the right

Apart from formal syntactic arguments, a word boundary-transcending dependency structure can also be motivated on purely practical grounds, since it will facilitate alignment, transfer and movement of semantic and functional equivalents between Greenlandic and other, more isolating languages in an MT context, and create a more comparable, deeper layer of syntax.

2 Morphosyntactic analysis

The input to our dependency grammar comes from a morphosyntactic tagger for (West-)Greenlandic, incorporating a finite-state transducer (FST)³ for its morphological analysis and a Constraint Grammar (CG) -based disambiguator⁴ that also assigns shallow

² I.e. syntactic independence of internal word parts

³ Online at: <https://oqaasileriffik.gl/sprogetknologi/lookup/?lookup=oqaasileriffik&meta=>

⁴ Both the FST and the CG grammar were originally developed by Per Langgård and his team at the Language Secretariat of Greenland (<https://oqaasileriffik.gl>), and continue to be actively developed, for instance for use in

syntactic function markers.

For instance, in the 3-word sentence below (*Anda tungujorumiK tujuulussivoq*), FST analysis provides a 6-way ambiguity for the second word, covering both verbal participle (TUQ derivation) and adjectival noun readings (no derivation) in both instrumental (Ins) and two relative (Rel) possessum (Poss) inflections.

Anda (Anda)

Anda+Sem/Mask+Prop+Abs+Sg

tungujortumik (blue)

tungujor+IV+TUQ+vn+N+Ins+Sg

tungujor+IV+TUQ+vn+N+Rel+Pl+4PlPoss

tungujor+IV+TUQ+vn+N+Rel+Sg+4PlPoss

tungujortoq+N+Ins+Sg

tungujortoq+N+Rel+Pl+4PlPoss

tungujortoq+N+Rel+Sg+4PlPoss

tujuulussivoq (sweater-buys/bought)

tujuuluk+SI+nv+V+Ind+3Sg

In the disambiguated sentence, in CG format, only one (adjectival noun) reading survives, and function tags are added for subject (@SUBJ>), predicate (@PRED) and modifier (@i->N).

Anda (Anda)

[Anda] Prop Abs Sg @SUBJ>

tungujortumik (blue)

[tungujortoq] N Ins Sg @i->N

tujuulussivoq (sweater-buys/bought)

[tujuuluk] SI+nv V Ind 3Sg @PRED

3 Extended dependency trees

3.1 Syntactic tokenization

In syntactic terms, especially comparative cross-language syntax, even the short Greenlandic sentence above contains two major challenges. First, in the unadapted system, with a standard CG tag set, the modifier tag on *tungujortumik* would have to be either @>N (prenominal) or @ADVL> (adverbial), but neither would be especially satisfactory, since the former lacks a surface-syntactic noun as a head (so no tree can be built), and the latter does match an existing head type (verb), but does not express the words true, attributive function. Second, the predicate

spell checking and machine translation.

verb actually incorporates its own object (*sweater*), with the verb *SI* (*buy*) added as a nomino-verbal affix (nv), a common phenomenon in Greenlandic, but one that renders the (indefinite) objects invisible in a standard tree structure.

Motivated by a bilingual MT perspective, we introduced two descriptive modifications, one categorical, one structural, to resolve this conflict and arrive at a syntactic tree closer to a cross-lingual deep structure. The first change adds an i-prefix to syntactic functions whose dependency head is incorporated ("hidden") within another word. Thus, the tag @i->N is a variant of the prenominal @>N tag, but will no longer need a surface head noun to allow a well-formed syntactic tree. The second change concerns the core topic of this paper, breaking up Greenlandic words into meaningful parts and introducing syntactic functions and relations for these parts, hereby enabling the construction of a semantically more complete and syntactically more universal tree.

In the example sentence (fig. 1), there is one such syntactic fault line to consider — between the root *tujuuluk* (*sweater*) and the verbalizing affix *SI* (*buy*). In the tree notation below, #n->m means a dependency link from a daughter n to a head m.

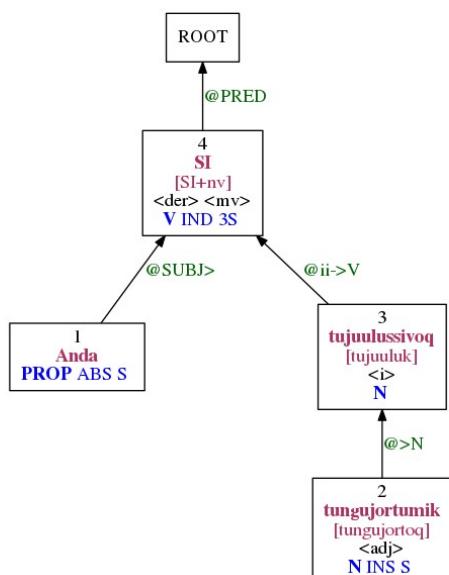


Fig. 1: Split-word dependency tree

Anda [Anda] (*Anda*)

PROP ABS S @SUBJ> #1->4

tungujortumik [tungujortoq] (*blue*)

<adj> N INS S @>N #2->3

tujuulussivoq [tujuuluk] (*a sweater*)

<i> N (S IDF) @ii->V #3->4

SI [SI+nv] (*buys/bought*)

<der> V IND 3S @PRED #4->0

Note that the prenominal function tag can now be standardized to @>N, as it now links to a "visible" noun entity with its own tree node (*tujuuluk*). The morphological cohesion between the parts of the erstwhile complex verb is maintained by inserting <i> tags (=internal) for all internal parts but the last, and <der> (=derivation) tags for all but the first. At the function level, we use dummy tags for word internal arguments, @ii->V for internal arguments of verbs, and @ii->N for internal arguments of nouns.

Modifiers and verb chain parts receive the same tags they would have had in ordinary CG. Consider the following 2-word sentence

timmisartumik [timmi] (*a plane*)

TAR+vv TUQ+vn N Ins Sg @MIK-OBJ>

titartaaniaanngilanga (*I didn't want to draw*)

[titartar] HTR+vv NIAR+vv NNGIT+vv V Ind 1Sg @PRED

After our dependency tree transformation, the auxiliary affix *NIAR* (*want*) as well as the light adverb *NNGIT* (*not*) will become tree nodes in their own right.

timmisartumik [timmisartoq] (*plane*)

N INS S @MIK-OBJ> #1->2

titartaaniaanngilanga [titartaavoq] (*draw*)

<HTR><i><mv> V @ii->V #2->3

NIAR [NIAR+vv] (*want*)

<der><i><hv><aux> V IND 1S @PRED #3->0

NNGIT [NNGIT+vv] (*not*)

<adv><der><tam> ADV @<ADVL #4->2

Note that the verbal inflection tags (V IND 1S) have been "raised" from their original position on the last affix to the auxiliary head verb, freeing the former to become an adverbial affix and allowing the latter to inherit the predicator (@PRED) and become top node of the sentence.

While splitting off of incorporated arguments, auxiliaries and light adverbs clearly pushes syntax under the water-line of the word boundary and helps to create a deeper syntax and a more universal dependency tree, there is also the danger of splitting off morphemes that are less syntactic in nature and part of larger semantic lexical units. For instance, in our example, the word for *plane* can be morphologically deconstructed into the root *timmi* (*plane*) and the affixes *TAR* (*uses to*) and *TUQ* (*that which*), literally meaning something (or somebody) that uses to fly. However, such a deconstruction is only of etymological interest, there are no external syntactic reasons for this (such as the existence of @i->V arguments), and the lexical minimal unit in terms of object equivalence in the real world is clearly *plane*. Similarly, the verb root *titartaavoq* (*draw*) is originally decomposed by the FST as *titartar(paa)+HTR*, i.e. with a transitive root and an affix denoting "half-transitivity" (i.e. taking an indefinite object in instrumental case). However, the *HTR* affix, while leaving morphological traces, does not correspond to a syntactic node, and since the external object is in an oblique case rather than ordinary object case (absolute), it syntactically "prefers" the longer and already half-transitive form *titartaavoq* as its dependency head (i.e. with *HTR* included).

3.2 Part-of-speech distribution

In a sense, our automatically performed word-splittings can be seen as a retokenization step turning Greenlandic into an orthographically more "normal" (i.e. not polysynthetic) language. When compared in terms of word class (POS) distribution, the two variants exhibited interesting differences, with the split Greenlandic version being closer to a Danish distribution⁵, a positive finding in the context of Machine Translation transfer alignment.

In table 1, percentages are drawn from an automatically annotated 9.1 million word corpus

⁵ For the Danish comparison, an annotated version of DSL's Korpus2000 was used, similar because of its high proportion of news text.

of Greenlandic news text⁶. All in all, the post-splitting corpus had 44.4% more tokens.

PoS	unsplit gl	split gl	not changed	first parts	da
N	54.4				
N n		37.4	24.3	5.7	21.2
N adj ⁷		4.9	2.1	-	6.7
N adv		2.2	2.0	-	
V	24.7				
V v		28.6	6.2	8.6	18.4
V adv ⁸		3.9	0.6	-	
V prp		0.8	0.8	-	13.1
ADV	3.7	2.6	2.2	~0	10.1
PROP	11.5	9.6	8.4	0.3	4.7
KC	1.4	3.6	0.8	~0	4.1
NUM	3.2	2.5	2.3	0.2	2.0
N num					
others*	1.1				19.3

Table 1: PoS percentages 80.7
*N(oun), V(erb), adv(erb), adj(ective), num(erical)
 PROP(er noun), KC=co-ordinating conjunction*

The original Greenlandic annotation is dominated by nouns (54%), but this is only because adjectives are regarded as nominal derivation of attributive verbs, and because non-finite clauses and relative clauses are expressed using nominal affixes (e.g TUQ and NIQ). In the retokenized corpus, the proportion between "semantic" nouns (N n) and "semantic" verbs (V v) is more balanced (1.3:1), close to the Danish proportion (1.2:1), with the difference in absolute numbers caused by the fact that a third of all Danish words are pronouns, prepositions and subordinators that have only inflectional equivalents in Greenlandic, meaning that Danish N and V counts would be 50% higher, if they would not have to share space with word classes

⁶ The corpus was compiled by Oqaasileriffik and will be made searchable at:
<https://tech.oqaasileriffik.gl/tools/corpus/>

⁷ adjectival "nouns" are morphologically ambiguous with relative clauses in Greenlandic, and in a split reading, the latter may be forced for syntactic reasons. Adjectival first parts remain invisible, because the lexicon forces a "be ADJ" verb root instead.

⁸ adverbial "verbs" come in two types: (a) Unsplit verbs in the contemporaneous mood functioning adverbially, and (b) adverbial affixes, typically last parts.

that do not exist in Greenlandic. For the minor word classes, too, after-splitting percentages are similar to those found for Danish⁹. The proper noun difference is due to the fact, that the Danish corpus regard multi-word names as tokens, while the Greenlandic tagged name parts individually.

3.3 Affix distribution

All in all, the fact that Greenlandic can be retokenized to match other languages' PoS distribution is typologically interesting and a strong argument for implementing such a tokenization in the face of bilingual tasks such as alignment and MT. In fact, the token-for-token similarity between retokenized Greenlandic and Danish becomes even more pronounced when looking at a more fine-grained affix distribution. Thus, the outer affixes in a Greenlandic verb, when read in inverse order from the verb end, nicely corresponds to a Danish chain of auxiliaries and light adverbs in the same order¹⁰, and even the auxiliary/verb proportion is similar (18.7% in Greenlandic, 21.1% in Danish).

About a quarter of all words were split, with each lexical first part spawning 1.78 split-off parts on average, or 2.05, when counting parts of dictionary-wise fused multiple affixes. Of these, 87% were affixes (88.8% when splitting multiple affixes), the rest enclitic particles (e.g. coordinating conjunctions). Verbo-verbal derivation was most common (+vv, 43.7%), cp. table 2:

	+ verbal affix	+ nominal affix
verb root	(vv) 43.7 %	(vn) 22.2 %
noun root	(nv) 19.4 %	(nn) 14.7 %

Table 2: root-affix pos combinations

From a top-17 list of individual affixes (table 3) it can be seen that a handful of heavily syntactic affixes are the most frequent ones,

covering in-word subclauses (NIQ, TUQ, TAQ), incorporated arguments (QAR, GE) and predicative-copula constructions (U, IP). The second most frequent are auxiliaries for passive (NIQAR), future (SSA(Q)), "aspect" (SIMA, TAR) and modality (SINNAA, NIAR), while there's only one adverb (NNGIT – not) and one real noun (VIK – place).

Affix	Grammar	%
NIQ+vn	nominal that/ing-clause	12.17
TUQ+nv	relative clause, adjectives attributive nouns	9.85
QAR+nv	have ROOT, there is ...	7.92
SSAQ+nn	future (of deverbal nouns)	7.88
NIQAR+vv	passive (aux)	6.94
IP+nv	copula	5.83
SSA+vv	future (of verbs, aux)	5.05
U+nv	copula	4.83
SIMA+vv	have ...ed, durative (aux)	4.66
TAR+vv	use to INF (habitually)	4.16
NNGIT+vv	negation (adverb)	3.34
SINNAA+vv	can (aux)	2.93
TIP+vv	make do, inchoative (aux)	2.49
TAQ+vn	relative clause passive	2.40
VIK+vn	place	2.28
GE+nv	have OBJ as ROOT	2.23
NIAR+vv	want to (aux)	1.77

Table 3: Affix distribution

4 Complex constructions

The following is a more complex example of a syntactic tree, with two subclauses (underlined) both expressed as single words in Greenlandic, but equivalent to 4-5 words in English or Danish:

Ilulissat Sermiat ukiumut 7 kilometerit
tikillugit sukkassuseqartoq sermip qanoq
sukkatigisumik ingerlaarsinnaaneran
takussutissaalluarpooq. – *The Ilulissat Glacier,
that has a speed reaching 7 km a year, is clearly
an indication of (the fact) how fast the ice can
move.*

As can be seen from the annotation (fig. 2), the first "clause-word" (*sukkassuseqartoq*) functions as a relative clause, where our algorithm splits off both the relative pronoun (TUQ) and the verb (QAR). However, a third affix, SSUSIQ (the quality of being ADJ), is *not* split off, because the (nominal) concept of an

⁹ For adverbs, this is true after lumping Greenlandic "inflectional" N/V adverbs together with "monolithic" adverbs and adverbs in the particle class (others).

¹⁰ e.g. *nerisinnannginnaku* (because I can't eat it)
neri+SINNAA+NNGIT+V-Cau-1Sg-3SgO
spise+kunne+ikke+fordi+jeg-det
eat+can+not+because-I-it

attribute (here: 'speed' = 'the quality of being fast') does constitute a purely semantic unit, without syntactic structure, a view that is supported by the fact that the concept of "speed" is recognized/realized as a word unit (rather than a construction) in many languages.

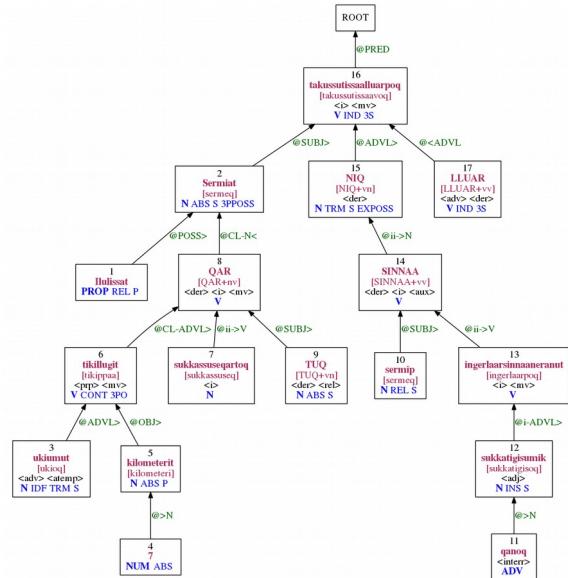


Fig 2: Complex dependency tree

Ilulissat [Ilulissat] (*Ilulissat*)
 PROP REL P @POSS > #1->2
 Sermiat [sermeq] (*Glacier*)
 N ABS S 3POSS @SUBJ > #2->16
 ukiumut [ukioq] (*per year*)
 N IDF TRM S @ADVL > #3->6
 7 [7] (*seven*) NUM ABS @>N #4->5
 kilometerit [kilometeri] (*kilometers*)
 N ABS P @OBJ > #5->6
 tikillugit [tikippaa] (*reaching / up to*)
 <pp> V CONT 3PO @CL-ADVL > #6->8
 sukkassuseqartoq [sukkassuseq] (*speed*)
 <SSUSIQ+vn> <i> N @ii->V #7->8
 QAR [QAR+vn] (*has*)
 <der> <i> <hv> <mv> V @CL-N <#8->2
 TUQ [TUQ+vn] (*that*)
 <der> <rel> N ABS S @SUBJ > #9->8
 sermip [sermeq] (*ice*)
 N REL S @SUBJ > #10->14
 qanoq [qanoq] (*how*)
 <interr> ADV @>N #11->12
 sukkatigisumik [sukkatigisoq] (*fast*)<adj><TIGE+vv>
 <TUQ+vn> N INS S @i-ADVL > #12->13
 ingerlaarsinnaaneranut [ingerlaarpoq] (*move*)
 <i> <mv> V @ii->V #13->14
 SINNAA [SINNAA+vv] (*can*)
 <der> <i> <hv> <aux> V @ii->N #14->15
 NIQ [NIQ+vn] (*the fact that*)
 <der> N TRM S EXPOSS @ADVL > #15->16
 takussutissaalluarpoq [takussutissaavoq] (*be an*

indication) <UTE+vn><SSAQ+nn> <U+nv> <i><mv><hv> V IND 3S @PRED #16->0
 LLUAR [LLUAR+vv] (*really*)
 <adv> <der> ADV @<ADVL #17->16

The second "clause-word" is a nominal (that-) clause, where the outermost affix (NIQ) can be said to replace the complementizer/conjunction in Germanic or Romance languages, while the verbal par, an auxiliary (SINNAA 'can') and the main verb (*ingerlaarpoq* – 'move') are incorporated. While a split here is clearly syntactic/structural and necessary for MT alignment, it does create a transformational problem: One constituent of the new subclause, the subject (*sermeq* 'ice') is inflected as a possessor (*sermip_REL*) and as such attaches to the whole (possessum-inflected) NIQ-noun, rather than its internal verb. In order to resolve this conflict, our grammar changes the function tag in the former (@SUBJ) and marks the possessum-inflection as EXPOSS in the latter. Both "clause-words" also have outside adverbial dependents, but these are marked as adverbial (or i-adverbial) even before retokenization, and do and not exhibit an adnominal morphology. Thus, *tikillugit* ('up to') is a verb in the comtemporative mood, typical of adverbial clauses or pp-heads, and *qanoq sukkatigisumik* ('how fast') does not have case agreement with the clausal NIQ-noun.

5 Annotation Procedure

In order to assign the dependency links discussed in the previous section, we use the CG3 formalism (Bick & Didriksen 2015), the same method that was originally used for disambiguating the morphosyntactic tags in our input. In this scheme, dependency links are assigned individually, from a target daughter token to a specified head type, using contextual conditions of arbitrary scope and complexity for both dependent and head independently. The following rule, for instance, handles nested possessor attachment.

SETPARENT @POSS > + S TO (*1 @POSS > - POSS BARRIER POSS/LU LINK pr POSS LINK *1A POSS + S BARRIER @POSS >);

The rule states that a possessor (@POSS >) in the

singular (S) attaches (TO) to a word inflected as a singular possessum (POSS + S), but it specifically targets the outer possessum in the nested structure, since it first looks right (*1) for another possessor without (BARRIER) a possessum or coordinator affix (LU) in between, then finds the inner possessor's already established parent to the right (pr) and finally attaches (A) to its own possessor, with a further BARRIER conditions for a possible third possessor. Ignoring further constituents, this will cover a construction like "Peter's having_eaten Anne's cake" which with a Greenlandic syntax would be "Peter's Anne's cake having_eaten":

(@POSS> #1->4 (((@POSS> #2->3 POSS @i-
ARG> #3->4) POSS #4->?))

All in all, our dependency grammar contains 251 such attachment rules and 319 other rules adapting existing function tags (e.g. the change from possessor to subject) or adding new ones for the split-off word parts. In addition, secondary tags are added, marking e.g. the individual parts of a coordination, or the verb functions of main verb, auxiliary and head verb¹¹.

Since our retokenization creates minimal syntactic tokens, the resulting Greenlandic dependency trees are much closer to the structure of Indo-European languages than the original annotation, facilitating machine translation into languages like English and Danish. Another interesting feature is the fact that most pronouns are only expressed in terms of verb inflection, and prepositions replaced by case marking. While this is a technical challenge to MT, it also makes for a small structural distance between ordinary syntactic trees and semantic trees (or tectogrammatical trees, as they are called in the Prague Dependency Treebank [Böhmová et al. 2003]). Thus, a future mark-up with semantic roles would not have to redraw the tree structure, because semantic heads are large equivalent to syntactic heads in (retokenized) Greenlandic.

6 Machine translation

With its lack of training data, its low-frequency

¹¹ Top/first/outermost verb of a verb chain

polysynthetic words and its difficult-to-align word-internal syntax, Greenlandic is a holdout for rule-based MT. Here, dependency annotation is a useful tool, if not a necessary prerequisite, for at least two important tasks, (a) lexical transfer and (b) syntactic transfer (Bick 2007). Thus, in a current MT initiative overseen by the Greenlandic Language Secretariat, contextual rules for the selection of translation equivalents can refer to morphosyntactic or semantic features of other tokens in the dependency tree: heads, dependents, siblings, granddaughter dependents etc. The transitive Greenlandic verb *suliaraa* ('to process'), for instance, translates into a number of different Danish verbs, depending on the semantic class (<...>) or lemma ("...") of its object (@OBJ) dependent (D):

suliaraa_V :behandle 'treat/process';
 * D=(<B.*> @OBJ) :dyrke 'grow'
 * D=(<(sem|cc-r).*> @OBJ) :udfærdige 'author'
 * D=(<act.*> @OBJ) :iværksætte 'launch'
 * D=("ameq" @OBJ) :garve 'tan'
 * D=("soraatummeerut") :besvare 'answer'

[=plant/botanical, <sem>=semiotic product, <cc-r>=readable object, <act>=action/activity]

Our syntactically motivated retokenization will allow translation selection conditions to "see" also affixes and incorporated arguments. Thus, head conditions for the adjectival noun *pikkunaatsoq* ('weak') will work even if the head noun is a verb-incorporated morpheme:

pikkunaatsoq_N <adj> :svag 'weak'
 * H=(<(cm-liq|drink)> :tynd, :vandet 'watery'
 * H=(<act>) :tam, :ineffektiv 'ineffective'
 * H=(<food.*>) :fad 'tasteless'

[<cm-liq>=liquid, <drink>=drink, <food>=food]

The other task involves movement of syntactic "treelets". For instance, in order to change (Greenlandic) SOV order into (Danish) SVO, object constituents have to be moved right, to a position after the vp. Given a dependency description, this can be expressed in one (simplified) rule, where a WITHCHILD condition means that the object token will be moved together with all its dependents and further descendants:

```

MOVE WITHCHILD (*) @OBJ
(NOT 0 <interr> OR <interr-head>)
AFTER WITHCHILD @MV< (pr <mv>) ;

```

(Move objects [@OBJ] with all () their children after a main verb <mv> dependency parent to the right (pr), but not if the object token in question is part of an interrogative np <interr>. The main verb constituent can include verb particles [@MV<].)*

Similarly, adjective phrases are moved from right to left within an np, and arguments of nouns (postnominal pp's in Danish) from left to right, etc. About 250 movement rules are needed for Greenlandic-Danish syntactic transfer.

7 Evaluation

In section 3, we have evaluated the quantitative impact of functional retokenization, and the resulting spread of affix types. However, in the absence of a gold corpus, or even a linguistic consensus as to how various Greenlandic constructions should look in a retokenized tree structure, it is difficult to do a classical recall/precision evaluation of the performance of the second step, dependency tagging. Still, it is reasonable to assume that morphosyntactic ambiguities and tagging failures in the input will affect the dependency layer. Thus, in a raw input run of the news corpus, 7.9% of non-punctuation tokens had no morphological analysis, though almost half of these could be heuristically tagged as proper nouns. Tokens that did have tagging had on average 1.13 readings (=13% ambiguity), and 3.2% had no syntactic function tag.

We addressed the missing-analysis problem with a post-processor that uses four different strategies for assigning heuristic analyses:

- (a) spell-checking (26%)
- (b) lexicalized dummy roots (13.2%)
- (c) rules for unknown proper nouns (15.3%)
- (d) endings-based heuristics (45.3%)

Together, these techniques covered almost all analysis failures and raised the syntactic coverage of the Greenlandic CG to 98.4%. The remaining 1.6% were assigned heuristic

functions in a postprocessing grammar, with 0.6% ending up with a dummy @X tag. It is a noteworthy consequence of the rich Greenlandic morphology that techniques (b) and (d) provided mostly correct POS and inflection (92.5%), and because syntactic function builds on case and mood inflection etc., it will also often be correct, at least at the unsplit level, even in the face of incorrectly suggested stems.

In order to approximate an evaluation of the dependency grammar in isolation, we presented it with input where all morphosyntactic tagging failures had been remedied heuristically. In this scenario, while possible errors would still carry over from the morphosyntactic annotation, the dependency grammar itself produced only 1.3% of formal errors, i.e. structurally unlikely or impossible dependency links. About 3/4 of these were unattached "orphan" tokens, 1/4 were type mismatches between daughter and head.

8 Conclusions and outlook

We have presented an affix-splitting dependency grammar module for a Greenlandic NLP pipe, implemented as a Constraint Grammar, with a special focus on MT, arguing for a syntactic treatment of non-inflectional morphemes. Our method increased the token count by 44.4% and led to a PoS distribution much more similar to that of the target language, Danish. In connection with a new heuristic strategy for morphosyntactic tagging failures, the dependency module identified formally acceptable dependency heads for 98-99% of tokens in retokenized CG input.

At the time of writing, the Greenlandic FST/CG tagger was still very much in flux in both descriptive and performance terms, but once it has stabilized, a gold standard dependency treebank for Greenlandic should be built allowing a better evaluation of the dependency tool. In the meantime, dependency annotation is still a very useful prerequisite for ML tasks such as context conditions in lexical transfer rules and syntactic movement rules.

References

- Bick, Eckhard; Tino Didriksen. 2015. CG-3 – Beyond Classical Constraint Grammar. In: Beáta Megyesi: *Proceedings of NODALIDA 2015, May 11-13, 2015, Vilnius, Lithuania.* pp. 31-39. Linköping: LiU Electronic Press. ISBN 978-91-7519-098-3
- Bick, Eckhard. 2007. Dan2eng: Wide-Coverage Danish-English Machine Translation, In: Bente Maegaard (ed.), *Proceedings of Machine Translation Summit XI, 10-14. Sept. 2007, Copenhagen, Denmark.* pp. 37-43
- Böhmová, Alena ; Jan Hajič; Eva Hajji; Barbora Hladká. 2003. The Prague Dependency Treebank: A Three-Level Annotation Scenario. In: Anne Abeillé (ed.): *Text, Speech and Language Technology Series*, Vol. 20. pp 103-127. Springer
- Compton, Richard; Pittman, Christine M. 2010. Word Formation by Phase in Inuit. *Lingua*, 120(9):2167-2192.
- Halle, M. & A. Marantz. 1993. Distributed Morphology and the Pieces of Inflection. In Hale, K. & S. J. Keyser (eds.): *The View from Building 20*. MIT Press, Cambridge, MA, pp. 111–176.
- Sadock, Jerrold M. 1980. Noun Incorporation in Greenlandic: A Case of Syntactic Word Formation . *Language*, Vol. 56, No. 2 (Jun., 1980), pp. 300-319. Linguistic Society of America

Metaphor detection for German Poetry

Ines Reinig

Computational Linguistics

Heidelberg University

wuerz@cl.uni-heidelberg.de

Ines Rehbein

Leibniz ScienceCampus

IDS Mannheim/Heidelberg University

rehbein@ids-mannheim.de

Abstract

This paper presents first steps towards metaphor detection in German poetry, in particular in expressionist poems. We create a dataset with adjective-noun pairs extracted from expressionist poems, manually annotated for metaphoricity. We discuss the annotation process and present models and experiments for metaphor detection where we investigate the impact of context and the domain dependence of the models.

1 Introduction

Metaphors are commonly used to conceptualise all aspects of our social and intellectual lives, thus helping us to make sense of the world around us (Lakoff, 1987, p.6). Therefore, many studies in NLP have addressed the task of metaphor detection for English and other languages, focussing on everyday language use. But metaphors are also an important stylistic device in literary texts, and recently more and more interest in computational methods for metaphor detection comes from the newly emerging area of Computational Humanities (Kesarwani et al., 2017; Tanasescu et al., 2018).

Our work is situated in the context of Computational Literary Studies. We are interested in the use of metaphors as stilistic devices in poetry, in particular in expressionist poems. Expressionism is an art movement originating in Germany at the beginning of the 20th century. In contrast to earlier periods such as Naturalism, expressionist artists focussed on describing the world not according to its physical properties but from a subjective and highly emotional perspective.

“Dem Dichter geht es also nicht um eine Darstellung der empirischen Wirklichkeit, sondern darum, wie er sie, nur er sie sieht und wie er möchte, daß sie auch von anderen gesehen werde. Er erarbeitet deshalb eine Metapher, die fähig ist, seine Gestimmtheit auszusprechen und

eine gleiche Gestimmtheit hervorzurufen: eine Art magische Formel.”¹ (Dietz, 1959, p.56)

For illustration, consider the following adjective-noun pairs from *Grodek*, a well-known expressionist war poem by Georg Trakl. In *Grodek*, Trakl creates a nightmarish atmosphere by means of colour symbolism, imagery, personification and neologisms, making extensive use of metaphors to express his inner view of reality (example 1).

- (1) a. *rotes Gewölk* (red clouds)
- b. *schwarze Verwesung* (black decay)
- c. *zerbrochene Miinder* (broken mouths)
- d. *wilde Klage* (wild lament)
- e. *schweigender Hain* (silent forest)
- f. *monde Kühle* (lunar coolness)

To be able to do large-scale investigations of metaphors in expressionist poems and to compare the use of metaphors in different literary genres or in the writings of individual authors, we need to be able to automatically detect metaphors in literary text with high precision and recall. This work presents first steps towards this goal. Our contributions can be summarised as follows:

- We create a new corpus with adjective-noun (A-N) pairs from expressionist poems, annotated for metaphoricity.
- We develop a classifier for automatically predicting A-N metaphors in literary texts.
- We investigate the domain dependence of our model by creating a second dataset for German A-N metaphors, based on the translation of the English A-N dataset of (Tsvetkov et al., 2014), extracted from web corpora.

¹Engl. translation: “The poet is not interested in a representation of empirical reality, but only in his subjective view of reality, and how he wants it to be seen by others. He therefore develops a metaphor that is capable of expressing his mood and evoking the same mood in others: a kind of magic formula.”

The paper is structured as follows. We first review related work on metaphor detection for English and German (§2). Then we describe the creation of the two datasets (§3) and present our experiments on metaphor detection for German (§4 and §5). We evaluate and discuss our results and outline avenues for future work (§6).

2 Related Work

Extensive research on metaphor detection has been conducted for English. Early approaches rely on lexical resources such as hyponym relations in WordNet and word co-occurrence information (Krishnakumaran and Zhu, 2007). Others have used abstractness ratings for individual words as features (Turney et al., 2011; Tsvetkov et al., 2014). Turney et al. (2011) show that abstractness scores extracted from a word’s context is an effective indicator of its metaphoricity. The system in Tsvetkov et al. (2014), which achieves an F-score of 85% on detecting English adjective-noun metaphors, uses imageability scores in addition to abstractness, in combination with WordNet supersenses and word embeddings.

Shutova et al. (2013) create a statistical model that does not depend on lexical knowledge from external knowledge bases but relies on weakly supervised distributional clustering. The more recent work in Rei et al. (2017) also identifies metaphors without the need for handcrafted features: a supervised similarity network uses the semantic information encoded in word embeddings to detect metaphorical relations. Their system is on a par with the work of Tsvetkov et al. (2014).

Only few studies have investigated metaphor detection for German, due to the lack of freely available annotated resources.² Köper and Schulte im Walde (2016a) develop a classifier for the identification of metaphorical uses of German particle verbs. Among other features, they use affective ratings for German lemmas (Köper and Schulte im Walde, 2016a) which we also employ in this work. Köper and Schulte im Walde (2017) model word senses for particle verbs and evaluate their model on metaphor detection, among other tasks.

3 Data & Annotation

In the paper, we focus on metaphorical adjective-noun (A-N) pairs and conduct experiments on two

datasets: i) one new dataset with A-N metaphors from German expressionist poems (POEMS) and ii) a second dataset based on a translation of the English A-N data of Tsvetkov et al. (2014) (TSV).

3.1 Annotating Metaphors in Poetry

For the first dataset, we extract A-N pairs from expressionist poems and annotate these pairs for metaphoricity. The process of creating and annotating the POEMS dataset is described below.

Dataset creation The poems have been collected from Project Gutenberg³, Deutsches Textarchiv⁴ and from various poetry websites. We extract the raw text and predict lemmas and POS tags using the TreeTagger (Schmid, 1994; Schmid, 1995). Then we extract lemma pairs that consist of an adjective followed by a noun.

In addition, we extract context for each A-N pair. Since the use of punctuation in poems does not always follow standard German grammar and sentence length in poetic texts can strongly vary in length, we choose to extract context information based on a fixed token window. For each A-N pair, we extract at most 10 tokens on the left and at most 10 tokens on the right. This approach generates context that varies only minimally in length.

We also limit the number of context strings extracted for each A-N pair to avoid that high-frequency A-N pairs are overrepresented in our data. For POEMS, we limit the number of context strings per pair to 20 in the training set and 10 in the test set. For the out-of-domain TSV dataset, we use a limit of 129 in the training set and 47 for the test set. These numbers were determined empirically such that i) no pair is overrepresented and ii) the original distributions between metaphorical and literal instances in the data is maintained.

Annotation procedure In the next step, we annotate each A-N pair with one of three labels (*literal*, *metaphorical*, *ambiguous*). The annotators do not see to the instance’s context but assign the label *ambiguous* for instances where context is necessary to disambiguate between literal and metaphorical uses. We found that most of the A-N pairs were unambiguous, making this procedure suitable for annotation and, at the same time, speeding up the annotation process by a large margin.

²The Hamburg Metaphor DB Project (Lönneker-Rodman, 2008) created a resource for French and (some) German metaphors. Unfortunately, the data is not publicly available.

³<https://gutenberg.spiegel.de>

⁴<http://www.deutsches-textarchiv.de/>

Example (2) shows an ambiguous instance from our corpus where the adjective *heiß* (hot) can refer to high temperatures in a literal sense or, metaphorically, to a subject of interest (hot topic). In such ambiguous cases, human annotators are usually able to determine the intended sense based on context. This was done in a second pass over the data where we presented the annotators with context for the ambiguous instances.

(2) *heißes Feld* (hot field)

Different approaches have been proposed for metaphor annotation. One of them is the Metaphor Identification Procedure (MIP) (Pragglejaz Group, 2007) which first establishes the contextual meaning of a lexical unit, then determines whether a more basic (concrete, precise, older or more related to bodily action) meaning exists. It then marks the unit as metaphorical if the contextual meaning contrasts with the basic meaning while being understandable in comparison with it.

A similar approach by Shutova (2017) uses the same definition of basic meaning but extends the annotation procedure by additionally identifying source and target domains. Shutova (2017) also highlights problems with the concept of *basic meaning*, i.e. the degree of conventionality of metaphors and the partially unsystematic inclusions of word senses in dictionaries make the use of dictionaries problematic for the identification of basic meanings. We encountered the same difficulties in the early stages of annotation when trying to use a dictionary as a reference. In consequence, we choose not to rely on dictionaries during the annotation process but instead extended our guidelines with a categorisation of adjectives and their interpretation (see A.2 in Appendix).

Do Dinh et al. (2018) address the problem of conventionalised metaphors by augmenting an English metaphor corpus with scores for metaphor novelty. They compare different approaches for annotation and show that best-worst scaling⁵, while being more time-consuming, yields the highest IAA. Their annotations, however, assume that the metaphors have already been identified.

Our annotation procedure follows previous work by marking A-N pairs as metaphorical if a more basic meaning of the adjective can be found. For example, in *durstiges Kind* (thirsty child), the adjective's meaning used to describe the noun can

⁵In best-worst scaling, annotators select the most novel and the most conventionalised from a set of four metaphors.

be considered as basic. In contrast, the adjective's meaning in *durstige Flamme* (thirsty flame) is considered to be different from the basic meaning.

The annotation is performed in several batches by two annotators. After each batch, the annotators discuss difficulties and annotation disagreements to discover grey areas not yet covered in the annotation guidelines, which were continuously improved during the annotation process.

While Tsvetkov et al. (2014) did not use context information in their experiments, we wanted to test the hypothesis that the context is useful for automatically distinguishing metaphors from literal senses. Therefore, after labeling each instance as either *metaphorical*, *literal* or *ambiguous*, annotators performed an additional annotation step and further annotated ambiguous A-N pairs as either metaphorical or literal by referring to their context. However, both annotators reported that they found this second step difficult because the context often did not provide enough information for disambiguation. Consider the following example:

- (3) Er schleudert die mächtigen [...] Kurven umher in der Welt, sie kehren zu ihm zurück, wie dem **dunklen Krieger**, der den Bumerang schnellt.

In the example above, *dunkler Krieger* (dark warrior) was labeled as ambiguous in the first round of annotation since *dunkel* (dark) could refer to colour (e.g. of the warrior's equipment or skin colour) in a literal sense, or to a gloomy or scary appearance in a metaphorical sense. Such ambiguities are characteristic for expressionist poems and again illustrate the use of metaphors as “a magic formula” (Dietz, 1959) to evoke certain emotions in readers.

As only 49 instances had been annotated as ambiguous, we decided to discard all ambiguous A-N pairs from the POEMS corpus, keeping only metaphorical and literal instances. All experiments described in section 5 are conducted on this two-class dataset.⁶

3.2 IAA and Error Analysis

We measured inter-annotator agreement (IAA) for the different batches during annotation. On average, we observe an IAA of 0.62 (Fleiss' κ) and a percentage agreement of 84.9%.

A particular challenge for annotation are adjectives of measurement. Take, for example, *hohe Kosten* (high costs) or *ein langer Tag* (a long day)

⁶We make all annotated data publicly available in form of lists (see <https://github.com/ireinig/metaphor-german-poetry/tree/master/data>).

where it is not clear whether the adjective’s basic sense should only refer to physical objects with spatial extensions (length, width, depth, height) or also capture other measures such as monetary values or the length of time. During annotation, we discussed these disagreements and extended the guidelines accordingly. For example, in the case of *groß* (big/large), we decided to mark instances as literal when the adjective refers to a quantifiable or classifiable attribute, such as size, surface or intensity, and label all other uses as metaphorically.

While most disagreements concern adjectives of measurement, we did not observe an annotation bias in terms of one annotator choosing a particular class particularly more often than the other. The probability of choosing the literal class varies between 70-80% for both annotators across all batches.

3.3 Translating an English Metaphor Dataset

To investigate the domain dependence of our metaphor detection model, we create a second metaphor corpus based on the English dataset of Tsvetkov et al. (2014). The dataset was created manually using collections of metaphors from the web (training set) and sentences from the TenTen Web corpus (test set). The domains in this dataset range from economics to politics and sports, and are thus crucially different from our POEMS corpus.

We automatically translated the English A-N pairs to German using DeepL⁷. The set of translated instances was then cleaned up by removing i) instances that are not A-N pairs (e.g. English A-N pairs translated to German N-N compounds) and ii) duplicate instances, resulting from the translation of two distinct English instances to the same German expression (e.g. *little chance* and *slim chance* were both translated to *geringe Chance*).

We then lemmatise the translated A-N pairs using the TreeTagger and extract context for each A-N pair from the sDeWaC German Web corpus (Faaß and Eckart, 2013). Table 1 shows the size of the dataset for the original English data (Tsvetkov et al., 2014) and for the translated Tsv dataset.

4 Experimental Setup

Training/test split We divide the data into training and test sets by putting all A-N pairs that appeared at least twice in the corpus in the training set while instances occurring only once constitute the

⁷<https://www.deepl.com/en/translator>

Lang	Set	Total	metaphorical	literal
POEMS dataset				
DE	Training	578	100	478
DE	Test	378	98	280
TSV dataset				
EN	train	1768	884	884
EN	test	200	100	100
DE	train	1149	546	603
DE	test	142	65	77

Table 1: Number of A-N pairs in the German POEMS and the English and German TSV datasets.

test set.⁸ This ensures that none of the test instances have been seen during training. Table 1 illustrates the class imbalance in this dataset: approximately 17% (train) and 26% (test) of the instances are metaphorical while the majority class accounts for 83% and 74% of the data.

4.1 Features

In our experiments, we use the following features that have been shown to be beneficial for metaphor detection in the literature.

Word embeddings are dense vector representations that capture syntactic and semantic properties of words (Turian et al., 2010). Previous work has used embeddings for metaphor detection and reported high scores for baseline models that rely only on word embeddings as features (Tsvetkov et al., 2014; Bulat et al., 2017; Rei et al., 2017; Shutova et al., 2016).

For each A-N pair, we extract embeddings for the adjective and for the noun from the 100-dimensional SkipGram embeddings of Reimers et al. (2014).⁹ We average both vectors and obtain one 100-dimensional compositional embedding vector for each A-N pair.

Supersenses Our next feature uses the GermaNet (Hamp and Feldweg, 1997) supersense taxonomy for adjectives and nouns where word senses (and the associated lemma forms) are sorted into semantic fields (e.g. *Menge* (set), *Gesellschaft* (society) or *Koerperfunktion* (bodily functions)).¹⁰

⁸All parameter tuning was done in a cross-validation setup on the training set.

⁹The embeddings are available from https://www.informatik.tu-darmstadt.de/ukp/research_6/ukp_in_challenges/germeval_2014.

¹⁰For the list of supersenses, please refer to http://www.sfs.uni-tuebingen.de/GermaNet/germanet_structure.shtml#Tops

Following Tsvetkov et al. (2014), we construct feature vectors by calculating the degrees of membership for noun and adjective supersenses. GermaNet contains 16 distinct semantic fields for adjectives and 23 for nouns. We extract supersense features for each A-N pair as follows. For a given word, we count the number of synsets s it belongs to. Then, for each semantic field f , we count the number of synsets s_f from the set s that are related to f . Finally, for each f , we compute the resulting value by diving s_f by s . We thus obtain vectors of length 16 for adjectives and vectors of length 23 for nouns. The resulting 39-dimensional vector representation is a concatenation of both vectors.

Affective ratings Tsvetkov et al. (2014) and Turney et al. (2011) show that abstractness and imageability scores are useful features for metaphor detection. We use ratings for abstractness, imageability, arousal and valence published by Köper and Schulte im Walde (2016b). The dataset contains ratings for 351,617 German lemmas, in a range of 0 to 10. According to Köper and Schulte im Walde (2016b), *abstractness* characterises anything that cannot be perceived using our senses, as opposed to concreteness; *imageability* refers to words for which one can easily form a mental image; *arousal* refers to the intensity of the emotion linked to a word and *valence* describes whether positive or negative emotions are linked to the word.

We extract affective ratings for each adjective and each noun, resulting in an 8-dimensional vector representation. Words for which no rating is available are assigned the default value of 5.0.

5 Experiments

We investigate the following three hypotheses:

- H1** Supersenses, word embeddings and affective ratings are useful features for A-N metaphor detection in German poetry.
- H2** Context features extracted from the A-N pair’s surrounding text can further improve classification accuracy for metaphor detection.
- H3** Metaphors are not domain-dependent but a general cognitive phenomenon, thus supplementary out-of-domain training data can improve results for metaphor detection in poetry.

Setup We train two SVM models on our datasets, POEMS and Tsv. For model selection, we perform the following three steps:

1. Algorithm selection and hyperparameter tuning
2. Feature selection for A-N pairs
3. Feature selection for context features

Model and feature selection was done separately for the POEMS and Tsv datasets. We refer to the models trained on each dataset as POEMS and Tsv.

5.1 Model Selection

Following previous work (Turney et al., 2011; Tsvetkov et al., 2014; Bulat et al., 2017), we experiment with three ML algorithms, i) a Random Forest classifier (Breiman, 2001), ii) a Support Vector Machine (SVM) (Joachims, 1998) and iii) logistic regression (Le Cessie and Van Houwelingen, 1992).¹¹ Based on 10-fold cross-validation on the training set, we select the SVM as the best performing model for the POEMS and Tsv datasets. We will use this model in all further experiments.

5.2 Class Imbalance in the POEMS Data

As shown in Table 1, the POEMS dataset is highly imbalanced, with far more instances for the non-metaphorical class. A common problem when training classifiers on imbalanced data is the classifier’s bias towards the majority class. Several techniques have been proposed to tackle this problem (Chawla, 2010). One example are resampling techniques where, in the case of *oversampling*, the minority class is increased by randomly adding duplicates from this class to the training set. *Undersampling*, on the other hand, reduces the number of instances from the majority class in order to obtain a more balanced distribution, at the cost of decreasing the size of the training data.

Another solution is *cost-sensitive learning* where the model is punished harder when misclassifying instances from the minority class while prediction errors on the majority class do not lead to high costs. We determine the best suited approach to deal with class imbalance using 10-fold cross-validation on the POEMS training set. We select a cost-sensitive SVM¹² with a Radial Basis Function (RBF) kernel for the POEMS model. We use this cost-sensitive model in all further experiments.

¹¹We use the Scikit-learn toolkit (Pedregosa et al., 2011) implementations for all models.

¹²In Scikit-learn, this algorithm can be made cost-sensitive by adapting the parameter `class_weight`, which controls the weights attributed to each class.

Features	F1 (macro)	stdev	F1 (M)	stdev
All features	72.7	(8.2)	54.7	(14.1)
All - supers.	70.2	(8.8)	51.3	(14.2)
All - embed.	<u>67.8</u>	(7.6)	<u>48.4</u>	(12.5)
All - ratings	71.9	(7.8)	53.6	(13.4)

Table 2: Feature ablation on the POEMS data (Macro F1 and F1(M): F1 for the minority class; stdev: standard deviation for cross-validation).

5.3 Feature Selection

We perform feature selection for POEMS and Tsv using feature ablation with 10-fold cross-validation on the respective training sets. We conduct these experiments to test our first hypothesis.

By dropping one feature at a time, we can determine the feature’s importance by measuring the decrease in performance in terms of F1-score. Table 2 shows 10-fold cross-validation results for POEMS. The highest F1-score is bolded while the lowest is underlined. Removing word embeddings results in the highest loss in performance, showing their usefulness for metaphor detection. Since we obtain highest performance when using all features, we conclude that all feature types contribute relevant information and keep them for the next set of experiments.

We conduct the same experiment on the Tsv data. As for the POEMS, best results are obtained when using all features. Results for the balanced Tsv dataset, however, are much higher with an F1-score of 82.8% (10-fold cross-validation on the training set).

We also compare the impact of different embeddings types. For POEMS, we obtain best results for the SkipGram embeddings (Reimers et al., 2014) while for Tsv, 100-dimensional FastText embeddings (Bojanowski et al., 2017) trained on the SDeWac corpus (Faaß and Eckart, 2013) give slightly higher results. We use FastText for all subsequent experiments on the Tsv dataset.

5.4 Context Features

Turney et al. (2011) state the hypothesis that ”the degree of abstractness of the context in which a given word appears is predictive of whether the word is used in a metaphorical or literal sense”. They support their claim with experiments showing that i) the abstractness of an adjective’s noun, seen as context, can be used to predict the adjective’s metaphoricity and ii) averaged abstractness ratings of a verb’s context, excluding the verb itself, can

be used to predict the verb’s metaphoricity. In all experiments, the authors report classification performances significantly higher than the majority class baselines and systems from related work.

While Turney et al. (2011) use only the noun modified by the adjective as context, we extend their hypothesis and test whether using features extracted from the A-N pairs’ surrounding context can further improve classification accuracy (**H2**). In addition to affective ratings, we also extract supersenses and word embeddings from the context and add these new features to the feature vectors.

Context feature extraction We extract features for word embeddings, supersenses and affective ratings from a context window of size 20 for each A-N pair and concatenate the additional feature representations with the feature vectors for the A-N pairs. Similar to Turney et al. (2011)’s experiments on verbs, we do not extract features for every word in the context but limit feature extraction to adjectives, nouns and verbs. The final context representation is the average over all individual context features for a specific A-N pair.

We use the same word embedding types that gave us best results in the previous experiments. In other words, we use Reimers et al. (2014)’s word embeddings for POEMS and FastText word embeddings trained on SDeWac for the TSV data.

Context feature selection We now compare the setting without context, using only features extracted for the A-N pairs, to models that are also trained on context features. Again, we perform an ablation study to measure each context feature’s importance, doing 5-fold cross-validation on the training set. The results for POEMS in Table 3 show a slight increase in overall F1-score; however, the improvement on the minority class is subtle and the high standard deviations for the different folds shows that the results are not robust. Since removing context supersenses from the set of context features lowers the performance, we test another setting in which we add only context supersense features to the feature set. This setting corresponds to the last row in Table 3. Since the cross-validation results suggest that context supersenses might include relevant information for the metaphor detection model, we add them to the feature vector.

We run the same experiment for Tsv. For this model, we achieve highest F1-scores without additional context features. This means that we found no evidence to support hypothesis **H2** that con-

Cont. features	F1 (macro)	stdev	F1 (M)	stdev
None	73.8	6.2	54.3	10.9
All	74.0	4.4	54.2	8.2
All - supers.	72.9	4.3	52.3	8.0
All - embed.	74.4	3.4	55.0	6.3
All - ratings	73.5	4.9	53.1	9.2
Only supers.	74.5	3.5	55.3	6.3

Table 3: POEMS context feature selection (in addition to features extracted from A-N pairs)

text features can provide useful information for metaphor detection for A-N pairs.

This is in contrast to Turney et al. (2011) who did report positive results for employing context features for metaphor detection. There are, however, some crucial differences between their and our setup. For adjectives, Turney et al. (2011) used only the adjectives’ nominal heads as context but did not include additional context features extracted from the local context of the A-N pair. Thus, their experiments only show improvements for using context for verbal metaphors where they do extract abstractness features for all nouns, adjectives and verbs in a sentence.

As a result, we do not know whether the use of additional context features might only be relevant for verbs where we would add information for verbal arguments that might be useful for disambiguation. For adjectives, we already include their syntactic heads in our setup and the surrounding context might not be as relevant as for verbal metaphors.

It might also be possible that our setup for extracting context information from a fixed-size window around the target A-N pair is suboptimal and that a different approach might be more successful. We leave this to future work.

Results on the test sets All experiments reported above were run in a cross-validation setup on the training sets and served to determine the best model and feature combinations for each dataset. We now report results on the test sets for the selected models and compare them to two baselines (Table 4). The *majority* baseline simulates a rule-based system that always predicts the majority class (i.e. literal) while *probability matching* corresponds to a classifier that makes random predictions according to the training set class distribution. For POEMS, we also report the performance using supersense context features in addition to the features extracted from A-N pairs (*All features + cont.*).

Model	Features	macro F1	F1 (M)
POEMS	Majority	42.6	0.0
	Probab. matching	53.2	26.3
	All features	62.9	42.8
	All + supersense cont.	61.7	37.1
Tsv	Majority	35.2	0.0
	Probab. matching	45.0	43.5
	All features	79.7	76.3

Table 4: Baselines and test set results for POEMS and Tsv. *All features* corresponds to the best model from the previous experiments.

For the POEMS, the features we investigate produce a model that substantially outperforms both baselines and seems to be able to distinguish between metaphorical and literal uses of adjectives. The improvements over the baselines, however, are not statistically significant. In addition, the supersense context features selected using cross-validation on the training set do not generalise to new data. We speculate that this might be related to the dataset’s class imbalance in addition to its relatively small size. Thus, increasing the size of the dataset is highly recommendable and should be the next step for future work.

For the balanced Tsv dataset, results for the selected model (last row of Table 4) are much higher with an average F1 of 79.7% and an F1 for the metaphorical class of 76.3%. Here, results are also significantly better than both baselines.¹³

5.5 Domain impact

In our last experiment, we test the usefulness of out-of-domain training data for metaphor detection in German poetry (**H3**).

The datasets used for POEMS and for Tsv present several differences: the former is unbalanced, as opposed to the latter, and of smaller size. Moreover, both datasets originate from different domains, namely i) poetry and ii) a range of different genres from the web. To test our hypothesis that out-of-domain data can be used to improve results for metaphor detection in poetry, we conduct the following experiment.

We merge the training sets from the Tsv and POEMS datasets and shuffle the resulting dataset with a fixed random state for reproducibility. We obtain a training set consisting of 624 metaphorical and 1,047 literal instances, which form a total of 1,617 training instances; we denote this new training set

¹³Using McNemar’s test, both p-values are below 0.000001.

Model	Training set	#	F1 (macro)	F1 (M)
Poems	Original	578	62.9	42.8
	Merged	1,671	58.8	32.9
TSV	Original	1,149	79.7	76.3
	Merged	1,671	81.2	78.3

Table 5: Test set results of POEMS and TSV using *merged* in comparison to original training sets

as *merged*. We then train the best models selected for POEMS and TSV on the *merged* training set and test the models on the original two test sets.

Table 5 shows results for the models trained on the *merged* training set; results using the original training sets are repeated for comparison. Adding out-of-domain training data to the POEMS did not improve results, meaning that we cannot confirm **H3** (repeated below).

H3: Metaphors are not domain-dependent but a general cognitive phenomenon, thus supplementary out-of-domain training data can improve results for metaphor detection in poetry.

The performance for the TSV data, however, did increase when adding the additional training data from the poetry corpus. At first glance, this is somewhat surprising as adding the TSV data to the poems results in a more balanced training set. This, however, might not be the best idea when the distribution in the test set is highly imbalanced. As a result, the classifier might have lost crucial information about the class distribution, which might explain the decrease in results.

To test whether this loss of information is responsible for the results, we run another experiment where we downsample both data sets so that both training and test sets have the same size and class distribution (table 6). Then we retrain and test our models on the resized datasets.¹⁴ Table 7 shows the same trend as before: training on POEMS does not decrease results for TSV while training on TSV yields substantially lower results for the POEMS.

¹⁴We report averaged results over 10 trials of sampling with replacement.

	train (M/L)	test (M/L)
POEMS (orig)	100/478	98/280
TSV (orig)	546/603	65/77
DOWN-SAMPLED	100/217	65/77

Table 6: Train size and distribution of Metaphorical / Literal instances in the datasets.

Model	Train-Test	Prec.	Rec.	F1	stdev
Poems	Poems-Poems	63.4	62.2	61.9	3.8
	Poems-TSV	75.0	65.9	64.1	2.0
TSV	TSV-TSV	75.1	65.2	63.0	2.3
	TSV-Poems	59.5	52.9	44.3	3.6

Table 7: Cross-domain results for downsampled datasets (averaged over 10 runs).

These results should be taken with a grain of salt as the datasets are very small. In future work, we would like to validate our findings on larger data.

For now, we cannot confirm that the difference in class distribution in the training and test sets is the underlying reason for our negative results regarding H3. We thus have to assume that the use of metaphors in expressionist poetry is crucially different from the one in every-day life, as described by Dietz (1959).

6 Conclusions & Outlook

In the paper, we presented first steps towards metaphor detection in German literature, in particular, in expressionist poetry. We created two datasets with adjective-noun pairs, manually annotated for metaphoricity, and evaluated models for metaphor detection for German. Our results show that features that have been used for other languages work well for German, too, and that word embeddings in particular are valuable features. We tested whether additional context information can improve classification accuracy, with negative results. We also explored the domain-dependence for metaphor detection by adding supplementary out-of-domain training data. Here, the results were mixed and require future investigation.

One important finding of our work is that results for metaphor detection are highly dependent on the class distribution in the dataset, and that balanced corpora give overly optimistic and thus misleading results.

For future work, the most important step is to increase the size of the datasets and to add annotations for other metaphor types, such as verbal metaphors. Metaphor annotation is a challenging task where agreement between annotators is often low. As has been noted before (Shutova, 2017; Gibbs, 1984), metaphoricity can be seen as a continuum. Thus, it might be recommendable to annotate metaphors on a scale instead of categorising them into binary classes. This would allow annotators to capture shades of gray and give them more flexibility while avoiding arbitrary decisions.

Acknowledgments

This research has been partly conducted within the Leibniz Science Campus “Empirical Linguistics and Computational Modeling”, funded by the Leibniz Association under grant no. SAS-2015-IDS-LWC and by the Ministry of Science, Research, and Art (MWK) of the state of Baden-Württemberg.

References

- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.
- Luana Bulat, Stephen Clark, and Ekaterina Shutova. 2017. Modelling metaphor with attribute-based semantics. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 523–528.
- Nitesh V. Chawla. 2010. Data mining for imbalanced datasets: An overview. In *Data Mining and Knowledge Discovery Handbook*, 2nd ed., pages 875–886.
- Ludwig Dietz. 1959. *Die lyrische Form Georg Trakls*. Salzburg. Otto Müller.
- Erik-Lân Do Dinh, Hannah Wieland, and Iryna Gurevych. 2018. Weeding out conventionalized metaphors: A corpus of novel metaphor annotations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1424.
- Gertrud Faaß and Kerstin Eckart. 2013. SdWaC – A Corpus of Parsable Sentences from the Web. In *Language processing and knowledge in the Web*, pages 61–68. Springer.
- Raymond W. Jr. Gibbs. 1984. Literal Meaning and Psychological Theory. *Cognitive science*, 8(3):275–304.
- Birgit Hamp and Helmut Feldweg. 1997. GermaNet – a Lexical-Semantic Net for German. *Automatic information extraction and building of lexical semantic resources for NLP applications*.
- Thorsten Joachims. 1998. Text categorization with Support Vector Machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.
- Vaibhav Kesarwani, Diana Inkpen, Stan Szpakowicz, and Chris Tanasescu. 2017. Metaphor detection in a poetry corpus. In *The Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, LaTeCH 2017, pages 1–9.
- Maximilian Köper and Sabine Schulte im Walde. 2016a. Automatic semantic classification of german preposition types: Comparing hard and soft clustering approaches across features. In *The 54th Annual Meeting of the Association for Computational Linguistics*, ACL 2016.
- Maximilian Köper and Sabine Schulte im Walde. 2016b. Automatically Generated Affective Norms of Abstractness, Arousal, Imageability and Valence for 350 000 German Lemmas. In *LREC*.
- Maximilian Köper and Sabine Schulte im Walde. 2017. Applying Multi-Sense Embeddings for German Verbs to Determine Semantic Relatedness and to Detect Non-Literal Language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 535–542.
- Saisuresh Krishnakumaran and Xiaojin Zhu. 2007. Hunting elusive metaphors using lexical resources. In *Workshop on Computational Approaches to Figurative Language*, pages 13–20.
- George Lakoff. 1987. *Women, Fire, and Dangerous Things*. Chicago University Press.
- Saskia Le Cessie and Johannes C. Van Houwelingen. 1992. Ridge Estimators in Logistic Regression. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 41(1):191–201.
- Birte Lönneker-Rodman. 2008. The hamburg metaphor database project: issues in resource creation. *Language Resources and Evaluation*, 42(3):293–318.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830.
- Pragglejaz Group. 2007. MIP: A Method for Identifying Metaphorically Used Words in Discourse. *Metaphor and Symbol*, 22(1):1–39.
- Marek Rei, Luana Bulat, Douwe Kiela, and Ekaterina Shutova. 2017. Grasping the finer point: A supervised similarity network for metaphor detection. In *The 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2017, pages 1537–1546.
- Nils Reimers, Judith Eckle-Kohler, Carsten Schnober, Jungi Kim, and Iryna Gurevych. 2014. GermEval-2014: Nested Named Entity Recognition with Neural Networks. In Gertrud Faaß and Josef Ruppenhofer, editors, *Workshop Proceedings of the 12th Edition of the KONVENS Conference*, pages

117–120, Hildesheim, October. Universitätsverlag
Hildesheim.

Helmut Schmid. 1994. Probabilistic Part-Of-Speech Tagging Using Decision Trees. In *Proceedings of International Conference on New Methods in Language Processing, 1994*.

Helmut Schmid. 1995. Improvements in Part-Of-Speech Tagging with an application to German. *EAL SIGDAT work-shop, 1995*.

Ekaterina Shutova, Simone Teufel, and Anna Korhonen. 2013. Statistical metaphor processing. *Computational Linguistics*, 39(2):301–353.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black Holes and White Rabbits: Metaphor Identification with Visual Features. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 160–170.

Ekaterina Shutova. 2017. Annotation of Linguistic and Conceptual Metaphor. In *Handbook of Linguistic Annotation*, pages 1073–1100. Springer.

Chris Tanasescu, Vaibhav Kesarwani, and Diana Inkpen. 2018. Metaphor detection by deep learning and the place of poetic metaphor in digital humanities. In *The 31st International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018*, pages 122–127.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *The 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pages 248–258.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10*, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peter D. Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *The 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011*, pages 680–690.

A Appendices

A.1 List of authors for the POEMS dataset

The following table lists the authors of the poems included in the POEMS corpus, with the number of poems for each author.

Author	# poems	Author	# poems
Gottfried Benn	10	Johannes R. Becher	6
Julius Maria Becker	24	Frieda Bettingen	22
Ernst Blass	68	Paul Boldt	8
Theodor Däubler	62	Gerrit Engelke	66
Max Herrmann-Neisse	12	Georg Heym	87
Jakob van Hoddis	8	Oskar Kanehl	4
Georg Kulka	28	Else Lasker-Schueler	151
Heinrich Lersch	47	Alfred Lichtenstein	123
Oskar Loerke	45	Ernst Wilhelm Lotz	19
Ludwig Rubiner	21	Gustav Sack	65
Daniel Schiebeler	4	Ernst Stadler	116
August Stramm	71	Ernst Toller	5
Georg Trakl	91	Franz Werfel	35
Alfred Wolfenstein	2		

Authors of poems in the dataset POEMS

A.2 Categorisation for adjectives and annotation criteria

Category	Instances in training set	Literal ex.	Metaphorical ex.	Ambiguous ex.	Criteria
Temperature	heiß, kalt, kühl, lau, schwül, eisig, mild, vereist, warm	heißer Stern heißes Auge	heißer Träne, heißes Auge	heißes Feld	Literal meanings: temperature property of an object ('Teller'), body part ('Hand') or a substance ('Wind'). Metaphorical: unusual uses of such adjectives are those describing a noun that would not commonly have a temperature attribute (like an eye or a tear).
Color	blau, braun, bunt, gelb, vergilbt, grau, grün, purpur, rosig, rot, schwarz, weiß	blauer Abend, blaue Nacht, blaues Licht	blaue Seele, braune Stille, roter Duft	blauer Tag	Literal: the noun refers to an object that can have the property of a color. Abstract nouns such as 'soul' or concepts that cannot be touched such as 'odor' are metaphorical cases. In difficult cases such as 'night', 'day', 'evening', the noun does not refer to a physical object but since it can have the attribute of a color, it is considered literal.
Material	golden, kristallin, silbern, steinern, seiden	goldenes Auge, goldene Stufe, goldenes Bildnis, goldener Wald, goldene Luft, goldene Wolke, seidener Strumpf, silberne Hand	steinerne Geduld	goldener Tag	Literal: when the noun refers to a physical object such as an item. Difficulties occurred for the adjective 'golden'; the decision was made to exclude any sense that does not refer to the material gold or its tint/color from literal tags. Metaphorical: abstract concepts (idea, feeling). 'goldener Tag' can mean that the sky is tinted in a golden color (literal) or a day (time indication) was excellent in a certain way (metaphorical); thus the pair is ambiguous.
Texture, substance	blank, dicht, dormig, dumpf, hart, kahl, rostig, trüb, weich, zart, fest	blankes Gewehr, blanker Himmel, dichte Wiese, hartes Lager	hartes Licht, harte Luft	weicher Weg	Literal pairs contain an adjective describing a physical object's texture or substance: items, surfaces, body parts or certain locations ('Wiese'). Yet light cannot have a hard texture, for example, thus the corresponding pair is metaphorical.
Shape, volume, weight	dick, dünn, dürr, eng, rund, schmal, spitz, gewölbt, gezackt, hohl, leicht, offen, schwer, tief, weit	dünnestes Licht, dünnster Nebel, dürres Rohr, tiefe Wunde	schweres Lid, schwerer Schlaf, dürre Straße	-	Literal pairs contain an adjective describing a physical object's shape, volume and weight. However, 'dünn' can concretely describe the narrow shape of a ray of light and is tagged as literal. 'schweres Lid' describes a person's eyelid with the tendency to close; thus the adjective is used in a metaphorical way.
Size, length, height	groß, klein, lang, kurz, hoch, niedrig	großer Himmel, große Not, kleines Lied, kleine Stimme, langer Tag, lange Stunde	großes Leben, große Nacht, großer Tag, hohe Lust	hohes Gut, hohe Nacht	Literal: quantifiable or classifiable attribute ('groß': of high size, surface or intensity, 'hoch': placed high in space or high in quantity or intensity, 'lang', 'kurz', 'klein': an object's length or an event's duration). Any other sense is metaphorical.
Age, time	alt, erst, früh, jung, jetzt, neu, reif, ewig	altes Haus, alter Gott, erster Mensch, neuer Mensch, frühes Gedicht, ewige Nacht	alter Tag, junger Frühling, junge Kraft, ewiger Raum	-	Objects and living beings can be described as old or young/new. A distinction is made between 'new' and 'young', since the second one is reserved to living beings (thus 'jung Frühling' is metaphorical). In the case of 'ewiger Raum', the adjective's sense is modified to describe a surface or extension in space, thus the pair is metaphorical.
Light	dunkel, düster, finster, glühend, hell, schimmernd, strahlend	dunkler Baum, dunkles Wasser, dunkle Nacht	dunkle Frage, dunkle Stimme, finsterner Zorn, glühendes Blut	wald, Weg, düstere Bild, finstere Wasserr, gähnende Erde	Adjectives such as 'dunkel', 'düster' and 'finster' refer to a low light context in the literal sense. Any pair that makes use of the second sense (uncanny, sinister) is tagged as metaphorical. This category presents a high amount of ambiguous cases since both senses are often possible and context would be necessary to determine the adjective's sense.

Does BERT Make Any Sense?

Interpretable Word Sense Disambiguation with Contextualized Embeddings

Gregor Wiedemann¹ Steffen Remus¹

¹Language Technology Group
Department of Informatics
Universität Hamburg, Germany
 {gwiedemann, remus, biemann}
 @informatik.uni-hamburg.de

Avi Chawla² Chris Biemann¹

²Indian Institute of Technology (BHU)
Varanasi, India

avi.chawla.cse16@iitbhu.ac.in

Abstract

Contextualized word embeddings (CWE) such as provided by ELMo (Peters et al., 2018), Flair NLP (Akbik et al., 2018), or BERT (Devlin et al., 2019) are a major recent innovation in NLP. CWEs provide semantic vector representations of words depending on their respective context. Their advantage over static word embeddings has been shown for a number of tasks, such as text classification, sequence tagging, or machine translation. Since vectors of the same word type can vary depending on the respective context, they implicitly provide a model for word sense disambiguation (WSD). We introduce a simple but effective approach to WSD using a nearest neighbor classification on CWEs. We compare the performance of different CWE models for the task and can report improvements above the current state of the art for two standard WSD benchmark datasets. We further show that the pre-trained BERT model is able to place polysemic words into distinct ‘sense’ regions of the embedding space, while ELMo and Flair NLP do not seem to possess this ability.

1 Synonymy and Polysemy of Word Representations

Lexical semantics is characterized by a high degree of polysemy, i.e. the meaning of a word changes depending on the context in which it is currently used (Harris, 1954). Word Sense Disambiguation (WSD) is the task to identify the correct sense of the usage of a word from a (usually) fixed inventory of sense identifiers. For the English language, WordNet (Fellbaum, 1998) is the most commonly used sense inventory providing more than 200K word-sense pairs.

To train and evaluate WSD systems, a number of shared task datasets have been published in the SemEval workshop series. In the *lexical sample* task (Kilgarriff, 2001; Mihalcea et al., 2004), a training set and a test set is provided. The relatively large data contains one sense-annotated word per training/test instance. The *all-words* task (Edmonds and Cotton, 2001; Snyder and Palmer, 2004) only provides a small number of documents as test data where each ambiguous word is annotated with its sense. To facilitate the comparison of WSD systems, some efforts have been made to provide a comprehensive evaluation framework (Raganato et al., 2017), and to unify all publicly available datasets for the English language (Vial et al., 2018b).

WSD systems can be distinguished into three types — knowledge-based, supervised, and semi-supervised approaches. *Knowledge-based* systems utilize language resources such as dictionaries, thesauri and knowledge graphs to infer senses. *Supervised* approaches train a machine classifier to predict a sense given the target word and its context based on an annotated training data set. *Semi-supervised* approaches extend manually created training sets by large corpora of unlabeled data to improve WSD performance. All approaches rely on some way of context representation to predict the correct sense. Context is typically modeled via dictionary resources linked with senses, or as some feature vector obtained from a machine learning model.

A fundamental assumption in structuralist linguistics is the distinction between signifier and signified as introduced by Ferdinand de Saussure (Saussure, 2001) in the early 20th century. Computational linguistic approaches, when using character strings as the only representatives for word meaning, implicitly assume identity between signifier and signified. Different word senses are simply collapsed into the same string representation. In

this respect, word counting and dictionary-based approaches to analyze natural language texts have been criticized as pre-Saussurean (Pêcheux et al., 1995). In contrast, the distributional hypothesis not only states that meaning is dependent on context. It also states that words occurring in the same contexts tend to have a similar meaning (Harris, 1954). Hence, a more elegant way of representing meaning has been introduced by using the contexts of a word as an intermediate semantic representation that mediates between signifier and signified. For this, explicit vector representations, such as TF-IDF, or latent vector representations, with reduced dimensionality, have been widely used. Latent vector representations of words are commonly called word embeddings. They are fixed length vector representations, which are supposed to encode semantic properties. The seminal neural word embedding model Word2Vec (Mikolov et al., 2013), for instance, can be trained efficiently on billions of sentence contexts to obtain semantic vectors, one for each word type in the vocabulary. It allows synonymous terms to have similar vector representations that can be used for modeling virtually any downstream NLP task. Still, a polysemic term is represented by one single vector only, which represents all of its different senses in a collapsed fashion.

To capture polysemy as well, the idea of word embeddings has been extended to encode word sense embeddings. Neelakantan et al. (2014) first introduced a neural model to learn multiple embeddings for one word depending on different senses. The number of senses can be defined by a given parameter, or derived automatically in a non-parametric version of the model. However, employing sense embeddings in any downstream NLP task requires a reliable WSD system in an earlier stage to decide how to choose the appropriate embedding from the sense inventory.

Recent efforts to capture polysemy for word embeddings give up on the idea of a fixed word sense inventory. Contextualized word embeddings (CWE) do not only create one vector representation for each type in the vocabulary, they also produce distinct vectors for each token in a given context. The contextualized vector representation is supposed to represent word meaning and context information. This enables downstream tasks to actually distinguish the two levels of the signifier and the signified allowing for more realistic modeling

of natural language. The advantage of such contextually embedded token representations compared to static word embeddings has been shown for a number of tasks such as text classification (Zampieri et al., 2019) and sequence tagging (Akbik et al., 2018).

Contribution: We show that CWEs can be utilized directly to approach the WSD task due to their nature of providing distinct vector representations for the same token depending on its context. To learn the semantic capabilities of CWEs, we employ a simple, yet interpretable approach to WSD using a k -nearest neighbor classification (kNN) approach. We compare the performance of three different CWE models on four standard benchmark datasets. Our evaluation yields that not all contextualization approaches are equally effective in dealing with polysemy, and that the simple kNN approach suffers severely from sparsity in training datasets. Yet, by using kNN, we include provenance into our model, which allows to investigate the training sentences that lead to the classifier’s decision. Thus, we are able to study to what extent polysemy is captured by a specific contextualization approach. For two datasets, we are able to report new state-of-the-art (SOTA) results.

2 Related Work

2.1 Neural Word Sense Disambiguation

Several efforts have been made to induce different vectors for the multiplicity of senses a word can express. Bartunov et al. (2016), Neelakantan et al. (2014), or Rothe and Schütze (2015) induce so-called sense embeddings in a pre-training fashion. While Bartunov et al. (2016) induce sense embeddings in an unsupervised way and only fix the maximum number of senses per word, Rothe and Schütze (2015) require a pre-labeled sense inventory such as WordNet. Then, the sense embeddings are mapped to their corresponding synsets. Other approaches include the re-use of pre-trained word embeddings in order to induce new sense embeddings (Pelevina et al., 2016; Remus and Biemann, 2018). Panchenko et al. (2017) then also use induced sense embeddings for the downstream task of WSD. Camacho-Collados and Pilehvar (2018) provide an extensive overview of different word sense modeling approaches.

For WSD, (semi-)supervised approaches with recurrent neural network architectures represent the

current state of the art. Two major approaches were followed. First, Melamud et al. (2016) and Yuan et al. (2016), for instance, compute sentence context vectors for ambiguous target words. In the prediction phase, they select nearest neighbors of context vectors to determine the target word sense. Yuan et al. (2016) also use unlabeled sentences in a semi-supervised label propagation approach to overcome the sparse training data problem of the WSD task. Second, Kågebäck and Salomonsson (2016) employ a recurrent neural network to classify sense labels for an ambiguous target word given its surrounding sentence context. In contrast to earlier approaches, which relied on feature engineering (Taghipour and Ng, 2015), their architecture only uses pretrained GloVe word embeddings (Pennington et al., 2014) to achieve SOTA results on two English lexical sample datasets. For the all-words WSD task, Vial et al. (2018a) also employ a recurrent neural network. But instead of single target words, they sequentially classify sense labels for all tokens in a sentence. They also introduce an approach to collapse the sense vocabulary from WordNet to unambiguous hypersenses, which increases the label to sample ratio for each label, i.e. sense identifier. By training their network on the large sense annotated datasets SemCor (Miller et al., 1993) and the Princeton Annotated Gloss Corpus based on WordNet synset definitions (Fellbaum, 1998), they achieve the highest performance so far on most all-words WSD benchmarks. A similar architecture with an enhanced sense vocabulary compression was applied in (Vial et al., 2019), but instead of GloVe embeddings, BERT wordpiece embeddings (Devlin et al., 2019) are used as input for training. Especially the BERT embeddings further improved the performance yielding new state-of-the-art results.

2.2 Contextualized Word Embeddings

The idea of modeling sentence or context-level semantics together with word-level semantics proved to be a powerful innovation. For most downstream NLP tasks, CWEs drastically improved the performance of neural architectures compared to static word embeddings. However, the contextualization methodologies differ widely. We, thus, hypothesize that they are also very different in their ability to capture polysemy.

Like static word embeddings, CWEs are trained on large amounts of unlabeled data by some vari-

ant of language modeling. In our study, we investigate three most prominent and widely applied approaches: **Flair** (Akbik et al., 2018), **ELMo** (Peters et al., 2018), and **BERT** (Devlin et al., 2019).

Flair: For the contextualization provided in the *Flair NLP* framework, Akbik et al. (2018) take a static pre-trained word embedding vector, e.g. the GloVe word embeddings (Pennington et al., 2014), and concatenate two context vectors based on the left and right sentence context of the word to it. Context vectors are computed by two recurrent neural models, one character language model trained from left to right, one another from right to left. Their approach has been applied successfully especially for sequence tagging tasks such as named entity recognition and part-of-speech tagging.

ELMo: *Embeddings from language models* (ELMo) (Peters et al., 2018) approaches contextualization similar to Flair, but instead of two character language models, two stacked recurrent models for words are trained, again one left to right, and another right to left. For CWEs, outputs from the embedding layer, and the two bidirectional recurrent layers are not concatenated, but collapsed into one layer by a weighted, element-wise summation.

BERT: In contrast to the previous two approaches, *Bidirectional Encoder Representations from Transformers* (BERT) (Devlin et al., 2019) does not rely on the merging of two uni-directional recurrent language models with a (static) word embedding, but provides contextualized token embeddings in an end-to-end language model architecture. For this, a self-attention based transformer architecture is used, which, in combination with a masked language modeling target, allows to train the model seeing all left and right contexts of a target word at the same time. Self-attention and non-directionality of the language modeling task result in extraordinary performance gains compared to previous approaches.

According to the distributional hypothesis, if the same word regularly occurs in different, distinct contexts, we may assume polysemy of its meaning (Miller and Charles, 1991). Contextualized embeddings should be able to capture this property. In the following experiments, we investigate this hypothesis on the example of the introduced models.

	SE-2 (Tr)	SE-2 (Te)	SE-3 (Tr)	SE-3 (Te)	S7-T7 (coarse)	S7-T17 (fine)	SemCor	WNGT
#sentences	8,611	4,328	7,860	3,944	126	245	37,176	117,659
#CWEs	8,742	4,385	9,280	4520	455	6,118	230,558	1,126,459
#distinct words	313	233	57	57	327	1,177	20,589	147,306
#senses	783	620	285	260	371	3,054	33,732	206,941
avg #senses p. word	2.50	2.66	5.00	4.56	1.13	2.59	1.64	1.40
avg #CWEs p. word & sense	11.16	7.07	32.56	17.38	1.23	2.00	6.83	5.44
avg k'	2.75	-	7.63	-	-	-	3.16	2.98

Table 1: Properties of our datasets. For the test sets (Te), we do not report k' since they are not used as kNN training instances.

3 Nearest Neighbor Classification for WSD

We employ a rather simple approach to WSD using non-parametric nearest neighbor classification (kNN) to investigate the semantic capabilities of contextualized word embeddings. Compared to parametric classification approaches such as support vector machines or neural models, kNN has the advantage that we can directly investigate the training examples that lead to a certain classifier decision.

The kNN classification algorithm (Cover and Hart, 1967) assigns a plurality vote of a sample’s nearest labeled neighbors in its vicinity. In the most simple case, one-nearest neighbor, it predicts the label from the nearest training instance by some defined distance metric. Although complex weighting schemes for kNN exist, we stick to the simple non-parametric version of the algorithm to be able to better investigate the semantic properties of different contextualized embedding approaches.

As distance measure for kNN, we rely on cosine distance of the CWE vectors. Our approach considers only senses for a target word that have been observed during training. We call this approach localized nearest neighbor word sense disambiguation. We use spaCy¹ (Honnibal and Johnson, 2015) for pre-processing and the lemma of a word as the target word representation, e.g. ‘danced’, ‘dances’ and ‘dancing’ are mapped to the same lemma ‘dance’. Since BERT uses wordpieces, i.e. subword units of words instead of entire words or lemmas, we re-tokenize the lemmatized sentence and average all wordpiece CWEs that belong to the target word. Moreover, for the experiments with BERT embeddings², we follow the heuristic by Devlin et al. (2019) and concatenate the averaged wordpiece vectors of the last four layers.

¹<https://spacy.io/>

²We use the bert-large-uncased model.

We test different values for our single hyper-parameter $k \in \{1, \dots, 10, 50, 100, 500, 1000\}$. Like words in natural language, word senses follow a power-law distribution. Due to this, simple baseline approaches for WSD like the *most frequent sense (MFS)* baseline are rather high and hard to beat. Another effect of the skewed distribution are imbalanced training sets. Many senses described in WordNet only have one or two example sentences in the training sets, or are not present at all. This is severely problematic for larger k and the default implementation of kNN because of the majority class dominating the classification result. To deal with sense distribution imbalance, we modify the majority voting of kNN to $k' = \min(k, |V_s|)$ where V_s is the set of CWEs with the least frequent training examples for a given word sense s .

4 Datasets

We conduct our experiments with the help of four standard WSD evaluation sets, two lexical sample tasks and two all-words tasks. As lexical sample tasks, SensEval-2 (Kilgarriff, 2001, SE-2) and SensEval-3 (Mihalcea et al., 2004, SE-3) provide a training data set and test set each. The all-words tasks of SemEval 2007 Task 7 (Navigli et al., 2007, S7-T7) and Task 17 (Pradhan et al., 2007, S7-T17) solely comprise test data, both with a substantial overlap of their documents. The two sets differ in granularity: While ambiguous terms in Task 17 are annotated with one WordNet sense only, in Task 7 annotations are coarser clusters of highly similar WordNet senses. For training of the all-words tasks, we use *a*) the SemCor dataset (Miller et al., 1993), and *b*) the Princeton WordNet gloss corpus (WNGT) (Fellbaum, 1998) separately to investigate the influence of different training sets on our approach. For all experiments, we utilize the suggested datasets as provided by the UFSAC

Model	SE-2	SE-3	S7-T7 (coarse)		S7-T17 (fine)	
			SemCor	WNGT	SemCor	WNGT
Flair	65.27	68.75	69.24	78.68	45.92	50.99
ELMo	67.57	70.70	70.80	79.12	52.61	50.11
BERT	76.10	78.62	73.61	81.11	59.82	55.16

Table 2: kNN with $k = 1$ WSD performance (F1%). Best results for each testset are marked bold.

framework³ (Vial et al., 2018b), i.e. the respective training data. A concise overview of the data can be found in Table 1. From this, we can observe that the SE-2 and SE-3 training data sets, which were published along with the respective test sets, provide many more examples per word and sense than SemCor or WNGT.

5 Experimental Results

We conduct two experiments to determine whether contextualized word embeddings can solve the WSD task. In our first experiment, we compare different pre-trained embeddings with $k = 1$. In our second experiment, we test multiple values of k and the BERT pre-trained embeddings⁴ in order to estimate an optimal k . Further, we qualitatively examine the results to analyze, which cases can be typically solved by our approach and where it fails.

5.1 Contextualized Embeddings

To compare different CWE approaches, we use $k = 1$ nearest neighbor classification. Table 2 shows a high variance in performance. Simple kNN with ELMo as well as BERT embeddings beats the state of the art of the lexical sample task SE-2 (cp. Table 3). BERT also outperforms all others on the SE-3 task.

However, we observe a major performance drop of our approach for the two all-words WSD tasks in which no training data is provided along with the test set. For S7-T7 and S7-T17, the content and structure of the out-of-domain SemCor and WNGT training datasets differ drastically from those in the test data, which prevents yielding near state-of-the-art results. In fact, similarity of contextualized embeddings largely relies on semantically *and* structurally similar sentence contexts of polysemic target words. Hence, the more example sentences can be used for a sense, the higher are the chances

³Unification of Sense Annotated Corpora and Tools. We are using Version 2.1: <https://github.com/getalp/UFSAC>

⁴BERT performed best in experiment one.

k	SE-2	SE-3	S7-T7	S7-T17
1	76.10	78.62	81.11	59.82
2	76.10	78.62	81.11	59.82
3	76.52	79.66	80.94	59.38
4	76.52	79.55	80.94	59.82
5	76.43	79.79	81.07	60.27
6	76.43	79.81	81.07	60.27
7	76.50	80.02	81.03	60.49
8	76.50	79.86	81.03	60.49
9	76.40	79.97	81.03	60.49
10	76.40	80.12	81.03	60.49
50	76.43	79.66	81.11	60.94
100	76.43	79.63	81.20	60.71
500	76.38	79.63	81.11	60.71
1000	76.38	79.63	81.11	60.71
MFS	54.79	58.95	70.94	48.44
Kågebäck (2016)	<i>66.90</i>	<i>73.40</i>	-	-
Yuan et al. (2016)	-	-	84.30	63.70
Vial et al. (2018a)	-	-	86.02	66.81
Vial et al. (2019)	-	-	90.60	71.40

Table 3: Best kNN models vs. most frequent sense (MFS) and state of the art (F1%). Best results are bold, previous SOTA is in italics and our best results are underlined.

that a nearest neighbor expresses the same sense. As can be seen in Table 1, the SE-2 and SE-3 training datasets provide more CWEs for each word and sense, and our approach performs better with a growing number of CWEs, even with a higher average number of senses per word as is the case in SE-3.

Thus, we conclude that the nearest neighbor approach suffers specifically from data sparseness. The chances increase that aspects of similarity other than the sense of the target word in two compared sentence contexts drive the kNN decision. Moreover, CWEs actually do not organize well in spherically shaped form in the embedding space. Although senses might be actually separable, the non-parametric kNN classification is unable to learn a complex decision boundary focusing only on the most informative aspects of the CWE (Yuan et al., 2016, p. 4).

5.2 Nearest Neighbors

K-Optimization: To attenuate for noise in the training data, we optimize for k to obtain a more robust nearest neighbor classification. Table 3 shows our best results using the BERT embeddings along with results from related works. For SensEval-2

and SensEval-3, we achieve a new state-of-the-art result. We observe convergence with higher k values since our k' normalization heuristic is activated. For the S7-T*, we also achieve minor improvements with a higher k , but still drastically lack behind the state of the art.

Senses in CWE space: We investigate how well different CWE models encode information such as distinguishable senses in their vector space. Figure 1 shows T-SNE plots (van der Maaten and Hinton, 2008) of six different senses of the word “bank” in the SE-3 training dataset encoded by the three different CWE methods. For visualization purposes, we exclude senses with a frequency of less than two. The Flair embeddings hardly allow to distinguish any clusters as most senses are scattered across the entire plot. In the ELMo embedding space, the major senses are slightly more separated in different regions of the point cloud. Only in the BERT embedding space, some senses form clearly separable clusters. Also within the larger clusters, single senses are spread mostly in separated regions of the cluster. Hence, we conclude that BERT embeddings actually seem to encode some form of sense knowledge, which also explains why kNN can be successfully applied to them. Moreover, we can see why a more powerful parametric classification approach such as employed by Vial et al. (2019) is able to learn clear decision boundaries. Such clear decision boundaries seem to successfully solve the data sparseness issue of kNN.

Error analysis: From a qualitative inspection of true positive and false positive predictions, we are able to infer some semantic properties of the BERT embedding space and the used training corpora. Table 4 shows selected examples of polysemic words in different test sets, including their nearest neighbor from the respective training set.

Not only vocabulary overlap in the context as in ‘along the *bank* of the river’ and ‘along the *bank* of the river Greta’ (2) allows for correct predictions, but also semantic overlap as in ‘little earthy bank’ and ‘huge bank [of snow]’ (3). On the other hand, vocabulary overlap, as well as semantic relatedness as in ‘land bank’ (5) can lead to false predictions. Another interesting example for the latter is the confusion between ‘grass bank’ and ‘river bank’ (6) where the nearest neighbor sentence in the training set shares some military context with the target sentence. The correct sense (*bank%1:17:01::Sloping*

Land) and the predicted sense (*bank%1:17:00::A Long Ridge or Pile [of earth]*) share high semantic similarity, too. In this example, they might even be used interchangeably. Apparently this context yields higher similarity than any of the other training sentences containing ‘grass bank’ explicitly.

In Example (10), the targeted sense is an action, i.e. a verb sense, while the predicted sense is a noun, i.e. a different word class. In general, this could be easily fixed by restricting the classifier decision to the desired POS. However, while example (12) is still a false positive, it nicely shows that the simple kNN approach is able to distinguish senses by word class even though BERT never learned POS classes explicitly. This effect has been investigated in-depth by Jawahar et al. (2019), who found that each BERT layer learns different structural aspects of natural language. Example (12) also emphasizes the difficulty of distinguishing verb senses itself, i.e. the correct sense label in this example is *watch%2:39:00::look attentively* whereas the predicted label and the nearest neighbor is *watch%2:41:00::follow with the eyes or the mind; observe*. Verb senses in WordNet are very fine grained and thus harder to distinguish automatically and by humans, too.

5.3 Post-evaluation experiment

In order to address the issue of mixed POS senses, we run a further experiment, which restricts words to their lemma and their POS tag. Table 5 shows that including the POS restriction increases the F1 scores for S7-T7 and S7-T17. This can be explained by the number of different POS tags that can be found in the different corpora (c.f. Table 6). The results are more stable with respect to their relative performance, i.e. SemCor and WNGT reach comparable scores on S7-T17. Also, the results for SE-2 and SE-3 did not change drastically. This can be explained by the average number of POS tags a certain word is labeled with. This variety is much stronger in the S7-T* tasks compared to SE-*.

6 Conclusion

In this paper, we tested the semantic properties of contextualized word embeddings (CWEs) to address word sense disambiguation.⁵ To test their capabilities to distinguish different senses of a particular word, by placing their contextualized vector

⁵The source code of our experiments is publicly available at: <https://github.com/uhh-lt/bert-sense>

	Example sentence	Nearest neighbor
	SE-3 (train)	SE-3 (test)
(1)	President Aquino, admitting that the death of Ferdinand Marcos had sparked a wave of sympathy for the late dictator, urged Filipinos to stop weeping for the man who had laughed all the way to the bank [A Bank Building].	They must have been filled in at the bank [A Bank Building] either by Mr Hatton himself or else by the cashier who was attending to him.
(2)	Soon after setting off we came to a forested valley along the banks [Sloping Land] of the Gwaun.	In my own garden the twisted hazel, corylus avellana contorta, is underplanted with primroses, bluebells and wood anemones, for that is how I remember them growing, as they still do, along the banks [Sloping Land] of the rive Greta
(3)	In one direction only a little earthy bank [A Long Ridge] separates me from the edge of the ocean, while in the other the valley goes back for miles and miles.	The lake has been swept clean of snow by the wind, the sweepings making a huge bank [A Long Ridge] on our side that we have to negotiate.
(4)	However, it can be possible for the documents to be signed after you have sent a payment by cheque provided that you arrange for us to hold the cheque and not pay it into the bank [A Financial Institution] until we have received the signed deed of covenant.	The purpose of these stubs in a paying – in book is for the holder to have a record of the amount of money he had deposited in his bank [A Bank Building].
(5)	He continued: assuming current market conditions do not deteriorate further, the group, with conservative borrowings, a prime land bank [A Financial Institution] and a good forward sales position can look forward to another year of growth.	Crest Nicholson be the exception, not have much of a land bank [Supply or Stock] and rely on its skill in land buying.
(6)	The marine said, get down behind that grass bank [A Long Ridge], sir, and he immediately lobbed a Mills grenade into the river.	The guns were all along the river bank [Sloping Land] as far as I could see.
	SemCor	S7-T17
(7)	Some 30 balloon [Large Tough Nonrigid Bag] shows are held annually in the U.S., including the world's largest convocation of ersatz Phineas Foggs – the nine-day Albuquerque International Balloon Fiesta that attracts some 800,000 enthusiasts and more than 500 balloons, some of which are fetchingly shaped to resemble Carmen Miranda, Garfield or a 12-story-high condom.	Homes and factories and schools and a big wide federal highway, instead of peaceful corn to rest your eyes on while you tried to rest your heart, while you tried not to look at the balloon [Large Tough Nonrigid Bag] and the bandstand and the uniforms and the flash of the instruments.
(8)	The condom balloon [Large Tough Nonrigid Bag] was denied official entry status this year.	Just like the balloon [Large Tough Nonrigid Bag] would go up and you could sit all day and wish it would spring a leak or blow to hell up and burn and nothing like that would happen.
(9)	Starting with congressman Mario Biaggi (now serving a jail sentence [The Period of Time a Prisoner Is Imprisoned]), the company began a career of bribing federal, state and local public officials and those close to public officials, right up to and including E. Robert Wallach, close friend and adviser to former attorney general Ed Meese.	When authorities convicted him of practicing medicine without a license (he got off with a suspended sentence [The Period of Time a Prisoner Is Imprisoned] of three years because of his advanced age of 77), one of his victims was not around to testify: he was dead of cancer."
(10)	Americans it seems have followed Malcolm Forbes's hot-air lead and taken to balloon [To Ride in a Hot-Air Balloon] in a heady way.	Just like the balloon [Large Tough Nonrigid Bag] would go up and you could sit all day and wish it would spring a leak or blow to hell up and burn and nothing like that would happen.
(11)	Any question as to why an author would believe this plaintive, high-minded note of assurance is necessary is answered by reading this book [A Published Written Work] about sticky fingers and sweaty scammers.	But the book [A Written Version of a Play] is written around a somewhat dizzy cartoonist, and it has to be that way.
(12)	In between came lots of coffee drinking while watching [To Look Attentively] the balloons inflate and lots of standing around deciding who would fly in what balloon and in what order [...].	So Captain Jenks returned to his harbor post to watch [To Follow With the Eyes or the Mind; observe] the scouting plane put in five more appearances, and to feel the certainty of this dread rising within him.

Table 4: Example predictions based on nearest neighbor sentences. The word in question is marked in boldface, subset with a short description of its WordNet synset (true positives green, false positives red).

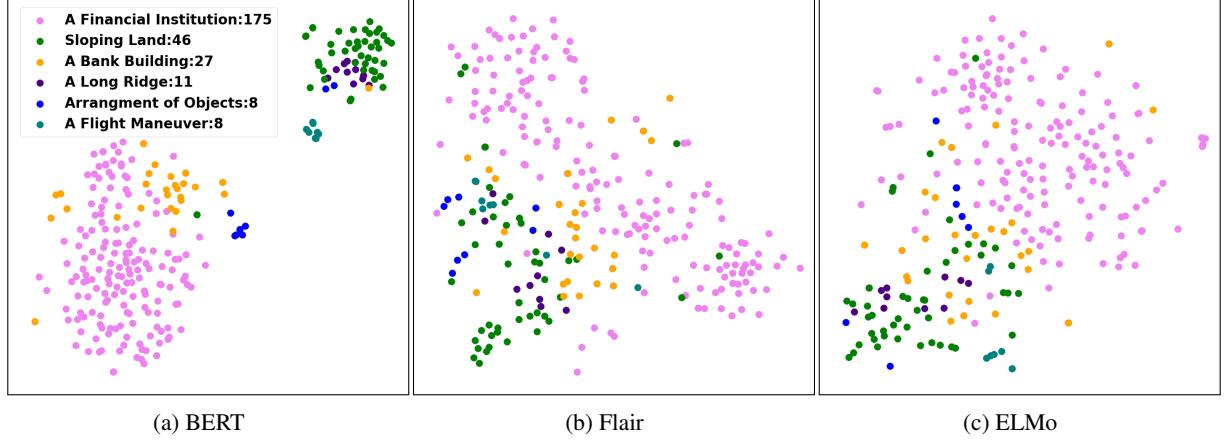


Figure 1: T-SNE plots of different senses of ‘bank’ and their contextualized embeddings. The legend shows a short description of the respective WordNet sense and the frequency of occurrence in the training data. Here, the SE-3 training dataset is used.

k	SE-2	SE-3	S7-T7		S7-T17	
			SemCor	WNGT	SemCor	WNGT
1	76.10	78.62	79.30	85.23	61.38	61.98
3	76.52	79.66	79.44	85.01	60.94	62.64
7	76.50	80.02	79.35	85.05	62.50	62.20
10	76.40	80.12	79.40	85.10	62.72	62.20
30	76.43	79.66	79.40	85.14	63.17	61.98
70	76.43	79.61	79.35	85.23	62.95	61.98
100	76.43	79.63	79.35	85.32	62.95	61.98
300	76.43	79.63	79.35	85.32	62.95	61.98

Table 5: Best POS-sensitive kNN models. Bold numbers are improved results over Table 3.

representation into different regions of the shared vector space, we used a k-nearest neighbor approach, which allows us to investigate their properties on an example basis. For experimentation, we used pre-trained models from Flair NLP (Akbik et al., 2018), ELMo (Peters et al., 2018), and BERT (Devlin et al., 2019). Further, we tested our hypothesis on four standard benchmark datasets for word sense disambiguation. We conclude that WSD can be surprisingly effective using solely CWEs. We are even able to report improvements over state-of-the-art results for the two lexical sample tasks of SenseEval-2 and SensEval-3.

Further, experiments showed that CWEs in general are able to capture senses, i.e. words, when used in a different sense, are placed in different regions. This effect appeared strongest using the BERT pre-trained model, where example instances even form clusters. This might give rise to future directions of investigation, e.g. unsupervised word sense-induction using clustering techniques.

	Noun	Adj	Verb	Other	avg #POS per word
SE-2 (tr)	41.32	16.57	42.11	0.00	1.0
SE-2 (te)	40.98	16.56	42.46	0.00	1.0
SE-3 (tr)	46.45	3.94	49.61	0.00	1.0
SE-3 (te)	46.17	3.98	49.86	0.00	1.0
S7-T7	49.00	15.75	26.14	9.11	1.03
S7-T17	34.95	0.00	65.05	0.00	1.01
SemCor	38.16	14.70	38.80	8.34	1.10
WNGT	57.93	21.57	15.55	4.96	1.07

Table 6: Percentage of senses with a certain POS tag in the corpora.

Since the publication of the BERT model, a number of extensions based on transformer architectures and language model pre-training have been released. In future work, we plan to evaluate also XLM (Lample and Conneau, 2019), RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019) with our approach. In our qualitative error analysis, we observed many near-misses, i.e. the target sense and the predicted sense are not particularly far away. We will investigate if more powerful classification algorithms for WSD based on contextualized embeddings are able to solve this issue even in cases of extremely sparse training data.

Acknowledgments

This work was funded by BWFG Hamburg in the Forum 4.0 project, by DFG in the JOIN-T 2 project and by DAAD in form of a WISE stipend.

References

- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, NM, USA.
- Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 130–138, Cadiz, Spain.
- Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63(1):743–788.
- Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, MN, USA.
- Philip Edmonds and Scott Cotton. 2001. SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, Toulouse, France.
- Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, USA.
- Zellig S. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.
- Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal.
- Ganesh Jawahar, Benoît Sagot, and Djamel Seddah. 2019. What does BERT learn about the structure of language? In *57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy.
- Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional LSTM. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 51–56, Osaka, Japan.
- Adam Kilgarriff. 2001. English lexical sample task description. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 17–20, Toulouse, France.
- Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.
- Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany.
- Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. 2004. The senseval-3 English lexical sample task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 25–28, Barcelona, Spain.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013*, Scottsdale, AZ, USA.
- George A. Miller and Walter G. Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.
- George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proceedings of the Workshop on Human Language Technology, HLT '93*, pages 303–308, Princeton, NJ, USA.
- Roberto Navigli, Kenneth C. Litkowski, and Orin Haragaves. 2007. SemEval-2007 task 07: Coarse-grained english all-words task. In *Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval '07*, pages 30–35, Prague, Czech Republic.
- Arvind Neelakantan, Jeevan Shankar, Alexandre Pasos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar.
- Alexander Panchenko, Fide Marten, Eugen Ruppert, Stefano Faralli, Dmitry Ustalov, Simone Paolo Ponzetto, and Chris Biemann. 2017. Unsupervised, knowledge-free, and interpretable word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 91–96, Copenhagen, Denmark.

- Michel Pêcheux, Tony Hak, and Niels Helsloot, editors. 1995. *Automatic discourse analysis*. Rodopi, Amsterdam and Atlanta.
- Maria Pelevina, Nikolay Arefiev, Chris Biemann, and Alexander Panchenko. 2016. Making sense of word embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 174–183, Berlin, Germany.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237, New Orleans, LA, USA.
- Sameer S. Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 task 17: English lexical sample, srl and all words. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 87–92, Prague, Czech Republic.
- Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110, Valencia, Spain.
- Steffen Remus and Chris Biemann. 2018. Retrofitting Word Representations for Unsupervised Sense Aware Word Similarities. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.
- Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China.
- Ferdinand de Saussure. 2001. *Grundfragen der allgemeinen Sprachwissenschaft*. de Gruyter, Berlin, 3 edition.
- Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain.
- Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 314–323, Denver, CO, USA.
- Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.
- Loïc Vial, Benjamin Lecouteux, and Didier Schwab. 2018a. Improving the coverage and the generalization ability of neural word sense disambiguation through hypernymy and hyponymy relationships. *CoRR*, abs/1811.00960.
- Loïc Vial, Benjamin Lecouteux, and Didier Schwab. 2018b. UFSAC: Unification of Sense Annotated Corpora and Tools. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.
- Loïc Vial, Benjamin Lecouteux, and Didier Schwab. 2019. Sense vocabulary compression through the semantic knowledge of wordnet for neural word sense disambiguation. *CoRR*, abs/1905.05677.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.
- Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1374–1385, Osaka, Japan.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, MN, USA.

Determining Response-generating Contexts on Microblogging Platforms

Jennifer Fest

RWTH Aachen University

English Linguistics

jennifer.fest@ifaar.rwth-aachen.de

Arndt Heilmann

RWTH Aachen University

English Linguistics

arndt.heilmann@ifaar.rwth-aachen.de

Oliver Hohlfeld

Brandenburg University of Technology

Computer Networks and Communication Systems

oliver.hohlfeld@b-tu.de

Stella Neumann

RWTH Aachen University

English Linguistics

stella.neumann@ifaar.rwth-aachen.de

Jens Helge Reelfs

RWTH Aachen University

Chair of Communication and Distributed Systems

helge.reelfs@comsys.rwth-aachen.de

Marco Schmitt

RWTH Aachen University

Department of Sociology

mschmitt@soziologie-rwth.aachen.de

Alina Vogelgesang

RWTH Aachen University

Department of Sociology

avogelgesang@soziologie.rwth-aachen.de

Abstract

In recent years the study of social media communities has come into the focus of research. One open but central question is which properties stimulate user interaction within communities and thus contribute to community building. In this paper, we provide a first step towards answering this question by identifying features in the *Jodel* microblogging app that trigger user responses as one form of attention. *Jodel* is a geographically restricted app that allows users to post threads and comments anonymously. The absence of displayed user information on *Jodel* makes the posted content the only trigger for user interaction, making the language the one and only means for users to gather contextual implications about their discourse partners. This enhanced function of language promises a revealing baseline investigation into linguistic behavior on social media.

To approach this issue, we conducted a sequence of lexico-grammatical analyses and

subjected the quantitative results to various statistical tests. While a Principal Component Analysis did not show a significant difference between the grammatical structure of original posts with and without answers, a negative binomial regression model focusing on the interpersonal meta-function yielded significant results. A further analysis of these features correlated to shorter or longer response times showed significant results for the interrogative mood. Additionally, keyword analyses revealed significant differences between posts with answers and without answers. Our study provides a promising first step towards understanding textual features triggering user interaction and thereby community building – an unresolved problem of practical relevance to social network operation.

1 Introduction

Social media in general and microblogging in particular create social spaces that are different from any we encounter in the physical world. In such

online spaces, we can communicate with others independently of geographical or social distance. This form of communication is typically enriched with meta information, e.g. user profiles, profile pictures or status messages. The social space takes on yet another form if posted content is not enriched by any data regarding the author – a new type of microblogging.

One such space is created by the app *Jodel*. It provides forums that are based solely on a user's location, i.e., only content in the user's proximity is shown. The users' feeds automatically change as soon as they move to a different location, forming individual local communities. As there is no way to use the app to intentionally get in touch with the same people repeatedly, the purpose cannot be to establish contacts or find specific like-minded users, but rather to participate in a local community/crowd. The only gain from posting on Jodel comes in the form of upvotes/downvotes and responses, i.e. in receiving attention in one way or another, something not every post is able to achieve. Many original posts (*original Jodels*, or *OJs*), pass by unnoticed, some generate only votes, others trigger lively and long-lasting discussions.

The absence of any user-related information makes the communication entirely anonymous and leaves the discourse partners to draw conclusions about each other on the basis of language only. Using Jodel and its content-specific form of communication therefore enables us to study language specific properties that trigger user interaction.

The aim of this study is to find out whether there are linguistic features which decide on or influence the success of a post in terms of it generating answers. We assume that certain topics, keywords and lexico-grammatical features trigger different response behavior patterns and intra-thread references; in some threads, we can mainly find references to the author of the OJ, while others entail active discussions during which participants refer to each other. These response-related patterns create networks that are structured very differently and, in future work, can work as a basis of comparing anonymous communication as on Jodel with non-anonymous discourse on platforms like Twitter or Facebook.

In the following sections, we will briefly review related work in the area of social media and introduce Jodel in some more detail. Section 4 will outline the dataset and methodology, before sec-

tion 5 will then present the results and conclusions drawn from them. As this is an exploratory study, we will outline future research perspectives, with a focus on transdisciplinary approaches and goals.

2 Research on Microblogging

Social media is subject to several lines of research. Generally, social media and the language that is used on respective platforms present a challenge to research in linguistics and communication science, as the discourse format is fairly recent and unusual and spelling and punctuation are less standardized (Golding et al., 2017). Analyses into this field require interdisciplinary approaches (Bouvier, 2015) as well as part of speech-tagging for highly non-standardized social media texts (Neunerdt, 2016). The most related lines of research can be summarized in four categories.

I) Firstly, the discourse patterns that are used on or caused by social media are a main focus. Conversational analyses have been conducted on Twitter (D'Heer and Verdegem, 2015; Freelon and Karpf, 2015) and Facebook (Androutsopoulos, 2015; Bolander and Locher, 2015), including also considerations on the function of such platforms as news hubs and multipliers (Cataldi et al., 2010). Many analyses in this field focus on individual events, like election campaigns (Enli, 2017), social or political movements (Poell, 2014; Kavada, 2015; Treré, 2015), natural disasters (Liu et al., 2016) or controversial criminal cases (McEnery et al., 2015); in short, events that often increase the use of social media and produce high frequency rates of the production of new content by the users.

Other studies focus on very concrete features of social or linguistic nature, but independent of any single (type of) event, thereby identifying characteristics that are more generalizable for social media. The topics here are very diverse and range from the creation of virtual friendships and groups (Wang et al., 2009; Chambers, 2013) to the linguistic manifestation of such affiliations (Zappavigna, 2012), dialectology (Eisenstein, 2018; Hovy and Purschke, 2018), humour (Locher and Bolander, 2015), language change and awareness (Dooly, 2018) and expectations towards peers (French and Bazarova, 2017), to name but a few.

II) Related, though not often explicitly connected to this strand are sociological investigations into the emergence of networks on social media platforms. Even more so than studies into linguis-

tic aspects, these analyses are almost exclusively limited to one platform, as their functions are too different to generalize findings or even permit the creation of similar network structures. Some, such as Facebook or Google+, work on the level of individuals as well as groups. They include fan-pages and promote the use of private chats and groups, and primarily make content visible to friends and contacts. Others, like Twitter or Instagram, rely on the profiles of individuals and, despite the function to ‘follow’ users, make content publicly visible. They are therefore used for different purposes, and network structures vary considerably. The dynamics of news production and trends, regardless of the topic, makes this a very interesting field of research.

III) Furthermore, as a special form of this dynamic structure, a number of studies analyses the implications of social media platforms on web 2.0. The vast majority of services does not provide a differentiation between laymen and experts, and all users can provide content and comment on events without being subject to moderation or quality control (Laux and Schmitt, 2017). Political, economic, scientific and mass media elites have discovered social media as a means of self-promotion and, as individuals or members of the organization, are clearly influencing the course of debates. While the tweets of regular users often have no (greater) resonance, well-known personalities attract a lot of attention and can use and increase social capital accumulated in other contexts. Reputation is mirrored in the number of ‘followers’ (Laux and Schmitt, 2017). This holds true for laymen as well, and social media platforms have become a major channel for aspiring models, musicians and other celebrities.

IV) Lastly, every online space has to cope with users that do not follow rules and produce fake news and abusive content. A lot of research is being conducted into creating means of identifying and filtering such content (Waseem and Hovy, 2016; Mondal et al., 2017; Baider, 2018; Ruzaite, 2018), and platform providers are eager for success in this field so as to avoid criticism and face potential consequences in the future. Especially since political parties and stakeholders have taken to social media and Donald Trump’s election in 2016 was accompanied by many accusations regarding “fake news”, this field of research has gained importance and has acquired an ideological dimension as

well. From an economic point of view, failing to prevent harmful content can seriously harm a platform and drive away users, as the downfall of the Jodel-alike app *YikYak* demonstrated quite clearly (Safronova, 2017).

In sum, popular platforms such as Twitter and Facebook are the dominant focus in research. On platforms like these, however, users always promote themselves to a certain degree and can be assumed to adapt their content and language to the image they wish to convey of themselves. Jodel, being anonymous and therefore useless for self-promotion, does not require its users to adapt to anything. The discourse here is much rawer and fewer variables interfere with or influence the language, making the analysis of this discourse a sound potential basis for research into other social media channels.

3 Jodel Explained

Jodel is a mobile-only microblogging app. Launched in 2014, it has been widely adopted in several European and the GCC countries. Like Twitter, Jodel enables users to share short posts of up to 250 characters and images (or short videos). Unlike Twitter and other traditional social media platforms, Jodel *a)* does not have user profiles, thereby making user to user communication anonymous (although users are enumerated within a thread, enabling interactions), and *b)* displays content only in the proximity of the user’s location, forming local communities.

Jodel is based on a community-driven filtering and moderation scheme to avoid adverse or abusive content. As stated above, effective moderation is a key parameter for the success of any social network or group messaging app. In Jodel, content filtering relies on a distributed voting scheme in which every user can raise or lower a post’s vote score by upvoting (+1) or downvoting (-1), similar to the mechanisms on other platforms such as StackOverflow or 9gag. OJs reaching a cumulative vote score below a negative threshold (e.g., -5) are not displayed anymore and can therefore not be commented or voted on any longer. Within threads, posts below this threshold are suppressed, but can still be clicked on to read, should anyone need their content to understand the whole discussion.

Depending on the number of vote contributions, this scheme filters out bad content while also potentially boosting mainstream content. As a second

line of defense, Jodel selects community moderators who have the authority to cast a moderator vote on posts that have been flagged (i.e. reported) by other users. On this superordinate level, the decision is again made by cumulative vote scores calculated when several moderators have made their choice. These moderators are supposed to decide on the basis of the app’s official guidelines, which forbids insults, sexually explicit pictures, any means of identification and other controversial content that can be clearly defined. Posts that are voted out on this level are automatically blocked entirely.

4 Method

For our analysis, the Jodel network operator provided us with anonymized data of their network. Our corpus (cf. Table 1) contains posts from the city of Aachen, Germany, from April 1 to August 31, 2017, as well as metadata on the individual posts. The focus on a single location and an arguably short time frame enabled us to focus on a more homogeneous group of users in the dimensions of content and time. All analyzed data has been publicly posted and thus been visible to all other Jodel users.

Metric	Entries
#Threads	182,413
#Responses	1,275,763
#Users	21,282
#Tokens	19,627,690

Table 1: Corpus

The corpus data was preprocessed with SoMeWeTa (Proisl, 2018), a part of speech-tagger for German social media and web texts. This tagger includes several tags specific to language used in social media texts in addition to the tags from the Stuttgart Tübingen Tag Set (STTS). Using a broad range of lexico-grammatical features as typically done in register analysis (Biber and Conrad, 2009; Neumann, 2014; Fest, 2016) allows us to explore linguistic variation in general, which might reflect differences between posts that begin a thread and receive answers from those that do not. These features range from lexical based measures (e.g., Lexical Density, Nominal Density) to approximations of clausal complexity (e.g., finite verbs per sentence). Frequency counts of these lexico-grammatical features were performed in the

Corpus Workbench (Evert and Hardie, 2011) with the help of a query script based on the example of a script developed by Neumann et al. (2017) for English and Dutch. The script exploits on part of speech-annotation in combination with positional information and word lists. Some queries address features specific to social media texts. The latter were queries for some of the additional tags provided by the SoMeWeTa, foreign language material, words specific to social media or the platform Jodel (*OJ*, *Heimatjodel*, etc.), colloquialisms and the metadata tags (hashtags, mentions, channels) used on Jodel.¹

In order to explain answer behaviour patterns and determine potential triggers of successfully generating responses, we used exploratory and confirmatory techniques. The first analysis is exploratory and is a global linguistic assessment of posts that start a thread (OJs) to find out if specific linguistic patterns emerge from a multitude of linguistic features that distinguish response-generating OJs from those that remain unanswered. This allows us to find out if successful OJs (OJs with at least one answer) are associated with particular sets of grammatical behavior(s). A total of 68 features was included after discarding collinear features ($r > |0.9|$). Each post is thus represented as a vector in a 68 dimensional space.

To reduce the dimensionality of the data, we first applied a Principal Component Analysis (PCA) to a random sample of the OJs in the Jodel corpus ($n=10,000$). The frequency results of the queries for the individual features were normalized with respect to appropriate units of measurement. For example, sentences were given per post, lexical measures such as nouns are included as the ratio of nouns to tokens, whereas passives and tense related features are given as the proportion of sentences. We also included measures of lexical density, which comes closest to a lexical indicator of the features considered. All values were standardized as z-scores, bringing the indicators to the same scale. Additionally, the indicators were transformed using a signed logarithm to reduce the skew of some of the variables that may have made it difficult to interpret the PCA. Based on these standardized indicators, a vector was built for each post which assumes a position in the multidimensional space.

In a second step, we shifted from the exploratory analysis to a confirmatory one and used regression

¹For a list of all queried features, see appendix A.

modelling with a set of linguistic features that we deemed reflective of social interaction. We decided to include first and second person pronouns, each per total amount of pronouns. Unfortunately, the second person plural pronoun *ihr* ('you' pl.) is indistinguishable from *ihr* ('her' sg.) in its third person singular sense, so there might be some overlap between both categories. We also included the number of imperatives, salutations and vocatives as well as the number of interrogatives per sentence. Additionally, we added the numbers of hashtags and emojis per total amount of tokens. Lastly, we controlled for number of words per sentence to account for length effects. Due to overdispersion issues with Poisson regression we used a negative binomial regression model. We hypothesized that it may also be specific content and not only linguistic behaviour per se, that triggers answers from the Jodel community. To this end, we conducted keyword analyses to filter out unusually frequent words in posts that gathered answers as opposed to those that did not.

5 Results

5.1 Grammatical Analysis

For the PCA, we assume that the Euclidean distances between feature vectors are suitable measures of (dis)similarity between data points (the OJs) with respect to the geometric configuration of vectors in multidimensional space and that these distances can be visualized using orthogonal projections to draw conclusions about the data. Clusters of posts that become apparent in the orthogonal projections can be interpreted as posts with different grammatical features. One or more clusters may be associated with grammatical features that attract answers.

Figure 1 visualizes the first three orthogonal dimensions. Most of the variance in the data is explained by the first dimension. Variance explained by the first three dimensions was 0.122, 0.058 and 0.052.

The PCA does not suggest a clear separation of OJs with and without answers. Separate clusters are virtually absent in the first three PCA dimensions and with respect to answers we could not identify a clear location of posts with no answer within the single big cluster.

According to Halliday et al. (2014), a significant part of the meaning potential of language revolves around interpersonal meanings, that is, around the

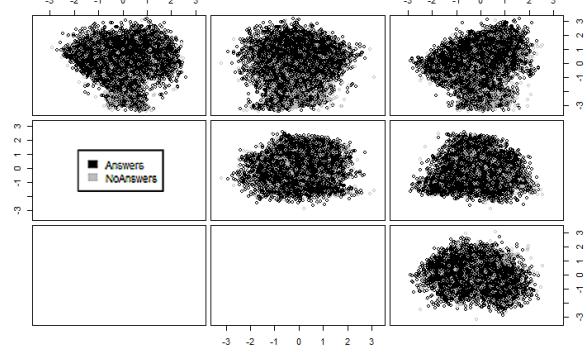


Figure 1: PCA Posts with answers / without answers

ways a writer uses to enact a social relationship with a reader. We hypothesized that features that can be associated with this kind of meaning could allow a more fine-grained understanding of linguistic features concerning the answer behavior of Jodel users. We therefore selected variables that we associated with indicators of dialogic interaction (cf. Table 2) and ran generalized linear regression models. This has the advantage that the effect of each variable on answering activity can be directly measured. We ran the regression models with the same 10,000 OJs but only a subset of the linguistic features from the prior analysis and predicted the number of answers.

No. of Answers	Estimate	Std. Error	z	p
<i>word_S</i>	0.024	0.002	10.97	<0.001
<i>emoji_T</i>	-0.374	4.174	-0.09	0.929
<i>hashtag_T</i>	-2.728	0.161	-16.979	<0.001
<i>p1pronoun_Pr</i>	0.364	0.048	7.59	<0.001
<i>p2pronoun_Pr</i>	-0.236	0.09	-2.632	<0.01
<i>ihr_Pr</i>	1.048	0.118	8.878	<0.001
<i>imperative_S</i>	-0.166	0.126	-1.315	0.188
<i>salutation_S</i>	0.184	0.129	1.434	0.152
<i>vocatives_S</i>	-0.095	0.211	-0.451	0.652
<i>interrogative_S</i>	0.377	0.043	8.766	<0.001

Table 2: Results for negative binomial regression model for the number of answers

The results show that several conversational features correlated positively and statistically significantly with the number of answers per OJ. These are words per sentence, interrogatives, first person pronouns, second person pronouns – particularly *ihr* and hashtags. Interestingly, an increased use of hashtags led to fewer responses, as did second person pronouns. Because we hold constant the effects for the second person plural *ihr*, this effect likely reflects the second person singular *du* ('you'). Why

Answer Delay	Estimate	Std. Error	z	p
<i>word_S</i>	-0.001	0.004	-0.155	0.877
<i>emoji_T</i>	1.894	9.693	0.195	0.845
<i>hashtag_T</i>	0.95	0.475	2.001	<0.05
<i>p1pronoun_Pr</i>	-0.032	0.097	-0.334	0.738
<i>p2pronoun_Pr</i>	-0.413	0.17	-2.43	<0.05
<i>ihr_Pr</i>	-0.151	0.235	-0.644	0.52
<i>imperative_S</i>	0.555	0.283	1.961	<0.05
<i>salutation_S</i>	-0.307	0.263	-1.169	0.243
<i>vocatives_S</i>	-0.268	0.428	-0.627	0.531
<i>interrogative_S</i>	0.228	0.087	2.614	<0.01

Table 3: Results for binomial logistic regression model for the time passed between OJ and the first answer (> 5 minutes resp. < 5 minutes). A positive sign indicates that a quick response is more likely than a slow response (e.g. interrogative mood makes a quick response more likely.)

an OJ would single out a specific addressee via *du* ('you' sg.) is somewhat interesting because Jodel is anonymous. A closer look at the data revealed that frequently this is used as an indefinite pronoun and often occurs in posts that reproduce dialogues of some kind or in reports of personal experience where the *du* actually refers to the OJ him-/herself and almost has the character of an internet meme:

“Wenn du morgens aufstehst, dich für die Uni fertig machst und losfährst und dann merkst, dass du anstatt zur Uni zu deiner alten Schule gefahren bist.” ('When you get up in the morning, get ready for university and start driving and then realize that you've driven to your old school instead of university.'

This kind of *du* thus refers to the OJ in the first instance and to others in a second instance, i.e. Jodel members that have made a similar experience or can relate to OJ. Syntactically it could be replaced with the pronoun *man* ('one' sg.), yet the achieved effect would be less personal. As a further analysis of the answer behavior we tested the time it took for the first person to answer the OJ with the same set of variables using a logistic binomial regression model. We categorized the answer time as "quick" (< 5 minutes) and "slow" (> 5 minutes) and predicted the likelihood of a quick answer compared to a slow answer. We chose to use a binomial model here because the model residuals were not normally distributed.

The model yielded significant results for interrogative mood, second person pronouns, imperatives and hashtags. The higher the ratio of hashtags per number of tokens, the more quickly a response

was issued – probably due to reasons of visibility. Second person pronouns lead to slower responses. In the light of the example above, the use of *du* may not actually invite responses (neither quantitatively nor temporally) and may rather be reflective of the OJ's need to express themselves. Typical conversational features that we also observe in face to face conversation (interrogative mood and imperative mood) were more likely to trigger quick responses. This is noteworthy because this is not face to face conversation but anonymous texting. The findings suggest that a variety of mood aspects elicit answers from the Jodel community and that asking questions and the use of hashtags trigger responses quickly. The latter, however, does not generate many answers at the same time.

5.2 Keyword Analysis

The last step of our investigation was keyword analyses to examine the content level and determine which topics are likely to generate answers on Jodel. Table 4 shows the keywords and their significance for OJs that generated answers in contrast to those that did not.

word	raw freq.	keyness ²	meaning/used as
<i>ihr</i>	18,099	907.91	pl. pronoun <i>you</i>
<i>jhj</i>	7,504	762.87	hashtag, <i>jodler helps jodler</i>
<i>oder</i>	11,949	462.89	<i>or</i>
<i>kann</i>	15,524	444.16	<i>can</i> , 3. p. sing.
<i>jemand</i>	13,124	373.00	<i>someone/anyone</i>
<i>habe</i>	11,410	295.32	<i>have</i> , 1. p. sing.
<i>wo</i>	6,041	256.10	<i>where</i>
<i>und</i>	53,899	254.31	<i>and</i>
<i>habt</i>	3,191	245.02	<i>have</i> , 2. p. pl.
<i>tipps</i>	1,296	242.77	<i>advice</i>

Table 4: Top 10 Keywords for Jodels with answers (vs Jodels without answers)

As can be seen, the hashtag *#jhj*, which signals a request for help from others, is one of the highest ranking keywords. The list furthermore includes *kann* in the 3rd person singular, which often collocates with *jemand*, indicating interrogatives. *Habt*, in 2nd person plural, is a direct address, and with *wo*, a direct question word is sixth in the list. The last item of the top ten keywords explicitly refers to advice.

Of those OJs which generate answers, we further examined the keywords for those that received the first answer within 5 minutes, in contrast to

²The keyness was calculated using Log-Likelihood and a threshold of $p < 0.05$

word	raw freq.	keyness	meaning/used as
<i>jhj</i>	6,512	886.35	hashtag, <i>jodler helps jodler</i>
<i>was</i>	14,293	150.73	<i>what</i>
<i>freundin</i>	3,548	125.64	(girl-)friend
<i>freund</i>	3,066	117.36	(boy-)friend
<i>frage</i>	2,138	103.53	question
<i>balloon</i>	231	103.26	emoji
<i>jodel</i>	4,109	90.45	
<i>er</i>	5,932	83.27	<i>he</i>
<i>warum</i>	2,600	81.38	<i>why</i>
<i>nicht</i>	18,776	81.34	<i>not</i>

Table 5: Top 10 Keywords for Jodels with an answer delay < 5 min. (vs Jodels with an answer delay > 5 min.)

those that had to wait longer. As Table 5 shows, direct questions appear even stronger here, with the interrogative pronouns *was*, *warum* and the noun *Frage* being key. Again, *#jhj* is strong, meaning that not only does this hashtag trigger responses as such, but fast ones, too. This is particularly interesting as hashtags in general, as was described above, have proven to be counterproductive as a discussion starter; *#jhj* appears to be a significant exception.

The results confirm the assumption that Jodel contains a certain service function showing itself in the active answering of questions other users might have.

Another interesting finding that can be gathered from the quick response keywords is that one topic shows to be particularly popular, which is relationships. *Freund* and *Freundin* are the only nouns apart from *Frage* in the ten strongest keywords. To further investigate this assumption, we split the OJs that had received answers into nine subgroups, based on how many answers they received (cf. Table 6).

OJs	42,419	27,367	30,648	23,845
#ANS	1-2	3-4	5-8	9-16
11,925	4,388	1,137	248	47
17-32	33-64	65-128	129-256	257-∞

Table 6: OJs with specific numbers of answers

As expected, the majority of OJs received only few answers; 42,419 (i.e. 30%) were answered only once or twice. Others were followed by long discussions of over 100 contributions. As an exploratory test, we contrasted the 1-2 answer-group with the other groups and then gradually moved

the border of contrast upwards. For a contrast of posts with up to 64 answers to those with 65 and more, only 19 keywords were significant at all for the long threads. 6 of these are directly linked to relationships and sex: *Männer* ('men'), *Beziehung* ('relationship'), *Kerle* ('guys'), *w* (for *weiblich*, 'female'), and *treu*, ('honest'/'faithful'). Another 6 are personal pronouns, and the emoji of the colourful rainbow, referring to LGBT communities, also features in the list.

5.3 Discussion

The results give some clear indications as to the formats and contents which most likely generate answers and discussions on Jodel for the Aachen community from April 2017 to August 2017, and also offer interesting insights into the particularities of the dynamics within an anonymous network. In contrast to other social media channels, where statements are made to produce reactions and users promote themselves and compete for followers, Jodel seems to be driven to a considerable degree by posting questions and getting answers. Regarding the popular topic of relationship and sex, it is likely that the anonymity on the platform is used to treat personal and sensitive issues which users would not discuss quite as openly elsewhere.

Because of its anonymity, the platform does not function as a public accumulation of status or social capital, in the form of likes or retweets, and the success of a post is not connected to a person or individual. Neither, of course, is failure of a post or disagreement to the posted content, which makes Jodel attractive to publish also controversial questions and opinions. The image of a person, which is at the center of interest on platforms like Twitter and Facebook, is of no consequence on Jodel and is therefore never at stake should anything not be received positively by the community.

Nevertheless, it can be seen that the Aachen Jodel users within our corpus indeed protect an image, if not their own. Questions are answered – and very often quickly – and especially calls for help generate fast responses. *#jhj* is such an example, and yet the answers to it are not always positive. The hashtag is key for posts that receive down-votes, particularly when the questions are very obviously either self-explanatory or could have been answered with little effort by the OJ. "Why don't you just google it?" or "Learn how to google!" are common answers in such instances, showing that users

take the time to reprimand someone although they have no immediate gain from that action, solely for the purpose of enforcing the community's rules and keeping up its identity. At this point, therefore, the attempt is made to maintain the image of the platform, part of which is that the questions asked should not be too trivial and that the content can be controversial, but should definitely be original and exciting.

Cultural and local identity is also mirrored in the topics that are received with most enthusiasm. Many general topics are discussed on the platform – from politics, university and job life, the city and parties to the weather and current events – and all of them receive various amounts of answers, but nothing sparks so much interest as relationships and sex. In Aachen, a city with a technical university and student population that is 68% male (RWTH Aachen University, 2018), how to get a partner, a one-night-stand or any other contact with the opposite sex are ever-present questions that are at times so dominant that the city's Jodel community has coined the term *geiern* ('to vulture') for males that hit on women all too obviously and persistently.

Taking all this into account, we can conclude that the anonymous character of Jodel leads to the creation of conventions – within the framework of the general user guidelines – at the center of which is not the identity of the user, but the image of a local community on the platform. This identity is cherished and protected, which in turn triggers an entirely unexpected and hard to define type of abusive content for a specific community, namely insulting anyone who does not abide by the self-imposed laws of the community.

6 Outlook and Future Work

As this last point already shows, an analysis into abusive content and hate speech is certainly promising on an anonymous platform. The community's identity is built and under constant negotiation, and users that offend the community's sense of self in some way have to expect to be discriminated against, even if they do not post anything objectively offensive. Finding and automatically filtering such instances is very difficult, yet we can assume that we can find similar mechanisms on other social media platforms as well. In future work, therefore, the understanding obtained by taking this focused perspective on the social network of Jodel can be extended to a broader set of locations and time

frames.

Some next steps within the present line of research are very apparent; by widening the dataset in a spatial and temporal dimension for a broader analysis, more complex models and classifiers as a predictor for a community's responsiveness can be developed. Also, we plan to extend our feature set to include metadata variables and examine more closely the role of hashtags and emojis.

As yet a different angle, a comparative approach to platforms like Facebook and Twitter seems promising. The differences in the contexts provided to the users by the platforms allow for the emergence of different social spaces, which in turn affects the purpose of using the platform as well as its communities and networks. This is reflected for instance in the use of hashtags. While a first look at hashtag usage on Jodel has shown that in many cases, such as the mentioned *#jhj*, hashtags define content of a category as they do on Twitter, many other hashtags on Jodel are means of content rather than of categorization (Fest et al., 2019; Reelfs et al., 2019). The negative effect of the hashtags on the likelihood of receiving answers indicates that this method, although frequently used, might not be too popular with the community.

On Twitter on the other hand, the function of hashtags is of a more exclusively categorizing nature, including the possibility to feature one's tweets in discussions on topics they are not related to simply by including the most trending hashtags of the moment. On both platforms, however, hashtags are a context marker – and the differences in usage therefore pose a window on just how context can be perceived and created.

Acknowledgement

This work has been funded by the Excellence Initiative of the German federal and state governments.

References

- Jannis Androutsopoulos. 2015. Networked multilingualism: Some language practices on Facebook and their implications. *International Journal of Bilingualism*, 19(2):185–205.
- Fabienne Baider. 2018. “Go to hell fucking faggots, may you die!” framing the LGBT subject in online comments. *Lodz Papers in Pragmatics*, 14(1):69–92.
- Douglas Biber and Susan Conrad. 2009. *Register*,

- Genre and Style*. Cambridge University Press, Cambridge, UK.
- Brook Bolander and Miriam A. Locher. 2015. "Peter is a dumb nut": Status updates and reactions to them as 'acts of positioning' in Facebook. *Pragmatics*, 25(1):99–122.
- Gwen Bouvier. 2015. What is a discourse approach to twitter, facebook, youtube and other social media: connecting with other academic fields? *Journal of Multicultural Discourses*, 10(2):149–162.
- Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. 2010. Emerging topic detection on Twitter based on temporal and social terms evaluation. In *Proceedings of the 10th International Workshop on Multimedia Data Mining*, page ch. 4. ACM, New York, NY.
- Deborah Chambers. 2013. *Social media and personal relationships: Online intimacies and networked friendship*. Palgrave Macmillan, Basingstoke.
- Evelien D'Heer and Pieter Verdegem. 2015. What social media data mean for audience studies: a multidimensional investigation of twitter use during a current affairs tv programme. *Information, Communication & Society*, 18(2):221–234.
- Melinda Dooly. 2018. "i h8 txt msgs". how social media has had an impact on language awareness. In Peter Garrett and Josep M. Cots, editors, *The Routledge Handbook of Language Awareness*, pages 306–322. Routledge, New York and London.
- Jacob Eisenstein. 2018. Identifying regional dialects in online social media. In Charles Boberg, Dominic James Landon Watt, and John A. Nerbonne, editors, *The handbook of Dialectology*, pages 368–383. John Wiley & Sons Inc, Hoboken, NJ.
- Gunn Enli. 2017. Twitter as arena for the authentic outsider: exploring the social media campaigns of trump and clinton in the 2016 us presidential election. *European Journal of Communication*, 32(1):50–61.
- Stefan Evert and Andrew Hardie. 2011. Twenty-first century Corpus Workbench: Updating a query architecture for the new millennium. In *Proceedings of the Corpus Linguistics Conference 2011, University of Birmingham, UK, 20-22 July 2011*, Birmingham, UK. University of Birmingham.
- Jennifer Fest, Stella Neumann, Arndt Heilmann, Oliver Hohlfeld, Jens Helge Reelfs, and Marco Schmitt. 2019. Hate speech dynamics in anonymous microblogging. In *29th European Systemic Functional Linguistics Conference*.
- Jennifer Fest. 2016. *News in the context of regional and functional variation: a corpus-based analysis of newspaper domains across varieties of English*. Phd thesis, RWTH Aachen University, Aachen, Germany.
- Deen Freelon and David Karpf. 2015. Of big birds and bayonets: Hybrid Twitter interactivity in the 2012 Presidential debates. *Information, Communication & Society*, 18(4):390–406.
- Megan French and Natalya N. Bazarova. 2017. Is Anybody Out There?: Understanding Masspersonal Communication Through Expectations for Response Across Social Media Platforms. *Journal of Computer-Mediated Communication*, 22(6):303–319.
- Peter Golding, Karin Raeymaekers, and Helena Sousa. 2017. Social media – new challenges and approaches for communications research. *European Journal of Communication*, 32(1):3–5.
- Michael Alexander Kirkwood Halliday, Christian Matthiessen, and Michael Halliday. 2014. *An introduction to functional grammar*. Routledge.
- Dirk Hovy and Christoph Puschke. 2018. Capturing regional variation with distributed place representations and geographic retrofitting. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4383–4394, Brussels, Belgium, October-November. Association for Computational Linguistics.
- Anastasia Kavada. 2015. Creating the collective: social media, the occupy movement and its constitution as a collective actor. *Information, Communication & Society*, 18(8):872–886.
- Henning Laux and Marco Schmitt. 2017. Der Fall Bautzen: Eine Netzwerkanalyse zur Entstehung digitaler Öffentlichkeiten. *Berliner Journal für Soziologie*, 27(3-4):485–520.
- Brooke Fisher Liu, Julia Daisy Fraustino, and Yan Jin. 2016. Social media use during disasters: How information form and source influence intended behavioral responses. *Communication Research*, 43(5):626–646.
- Miriam A. Locher and Brook Bolander. 2015. Humour in microblogging: Exploiting linguistic humour strategies for identity construction in two facebook focus groups. In Marta Dynel and Jan Chovanec, editors, *Participation in Public and Social Media Interactions*, volume 256 of *Pragmatics & Beyond New Series*, pages 135–155. John Benjamins, Amsterdam.
- Tony McEnery, Mark McGlashan, and Robbie Love. 2015. Press and social media reaction to ideologically inspired murder: The case of lee rigby. *Discourse & Communication*, 9(2):237–259.
- Mainack Mondal, Leandro Araújo Silva, and Fabrício Benevenuto. 2017. A Measurement Study of Hate Speech in Social Media. In Peter Dolog, Peter Vojtas, Francesco Bonchi, and Denis Helic, editors, *Proceedings of the 28th ACM Conference on Hypertext and Social Media - HT '17*, pages 85–94, New York. ACM Press.

- Stella Neumann, Gert De Sutter, and Stefan Evert. 2017. Register-specific interference in translation. In *Conference booklet of the 39th Annual Conference of the German Linguistic Society*, Saarbrücken, Germany, March.
- Stella Neumann. 2014. *Contrastive Register Variation: A Quantitative Approach to the Comparison of English and German*. De Gruyter Mouton, Berlin.
- Melanie Neunerdt. 2016. *Part-of-Speech tagging and detection of social media texts*. Apprimus, Aachen, Germany.
- Thomas Poell. 2014. Social media and the transformation of activist communication: exploring the social media ecology of the 2010 toronto g20 protests. *Information, Communication & Society*, 17(6):716–731.
- Thomas Proisl. 2018. SoMeWeTa: A Part-of-Speech Tagger for German Social Media and Web Texts. In Calzolari N, Choukri K, Cieri C, Declerck T, Goggi S, Hasida K, Isahara H, Maegaard B, Mariani J, Mazo H, Moreno A, Odijk J, Piperidis S, Tokunaga T, editor, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 665–670, Miyazaki. European Language Resources Association.
- Helge Reelfs, Timon Mohaupt, Oliver Hohlfeld, and Niklas Henckell. 2019. Hashtag usage in a geographically-local microblogging app. In *Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019.*, pages 919–927.
- Jurate Ruzaite. 2018. In search of hate speech in Lithuanian public discourse: A corpus-assisted analysis of online comments. *Lodz Papers in Pragmatics*, 14(1):93–116.
- RWTH Aachen University. 2018. Facts and figures.
- Valeriya Safranova. 2017. The Rise and Fall of Yik Yak, the Anonymous Messaging App. *The New York Times*.
- Emiliano Treré. 2015. Reclaiming, proclaiming, and maintaining collective identity in the #yosoy132 movement in mexico: an examination of digital frontstage and backstage activism through social media and instant messaging platforms. *Information, Communication & Society*, 18(8):901–915.
- Zuoming Wang, Joseph B. Walther, and Jeffrey T. Hancock. 2009. Social identification and interpersonal communication in computer-mediated communication: What you do versus who you are in virtual groups. *Human Communication Research*, 35(1):59–85.
- Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 88–93. ACL, Stroudsburg, PA.
- Michele Zappavigna. 2012. *The discourse of Twitter and social media*. Continuum, London and New York.

Appendix A. Queried linguistic features

Feature	Details
lexical_density	Number of lexical words divided by the number of tokens
nn_T	Number of nouns divided by the number of tokens
ne_T	Number of proper nouns divided by the number of tokens
nominal_T	Number of nominalizations divided by the number of tokens
neoclass_T	Number of neoclassical compounds divided by the number of tokens
poss_T	Number of possessive pronouns divided by the number of tokens
pronouns_T	Number of pronouns divided by the number of tokens
p1pronoun_Pr	Number of 1st person personal pronouns divided by the number of pronouns
p2pronoun_Pr	Number of 2nd person personal pronouns divided by the number of pronouns
p3pronoun_Pr	Number of 3rd person personal pronouns divided by the number of pronouns
ihr_Pr	Number of instances of the pronoun <i>ihr</i> divided by the number of pronouns
es_Pr	Number of instances of the pronoun <i>es</i> divided by the number of pronouns
pospers1_Pr	Number of all 1st person pronouns divided by the number of pronouns
pospers2_Pr	Number of all 2nd person pronouns divided by the number of pronouns
pospers3_Pr	Number of all 3rd person pronouns divided by the number of pronouns
adv_T	Number of adverbs divided by the number of tokens
adj_T	Number of adjectives divided by the number of tokens
atadj_T	Number of attributive adjectives divided by the number of tokens
prep_T	Number of prepositions divided by the number of tokens
finite_S	Number of finite verbs divided by the number of sentences
finite_V	Number of finite verbs divided by the number of verbs
pasttense_F	Number of past tense verbs divided by the number of finite verbs
perfect_F	Number of perfect verbs divided by the number of finite verbs
plusquamperfect_F	Number of instances of past perfect divided by the number of finite verbs
will_F	Number of instances of the modal verb <i>werden</i> used to signal future divided by the number of finite verbs
modalverb_V	Number of modal verbs divided by the number of verbs
verb_T	Number of verbs divided by the number of verbs
infinite_verbs_F	Number of infinitives with <i>zu</i> divided by the number of sentences
passive_S	Number of instances of passive voice divided by the number of sentences
coordination_S	Number of coordinating conjunctions divided by the number of sentences
subordination_S	Number of subordinating conjunctions divided by the number of sentences
interrogative_S	Number of instances of interrogative mood divided by the number of sentences
imperative_S	Number of instances of imperative mood divided by the number of sentences
politeimperative_S	Number of polite imperatives divided by the number of sentences
subjunctive_S	Number of modal verbs in subjunctive mood divided by the number of sentences
title_T	Number of titles divided by the number of tokens
salutation_S	Number of salutations and greetings (eg. <i>Hallo, Tschüss, Viele Grüße</i>) divided by the number of sentences
placeadv_T	Number of adverbs of place divided by the number of tokens
timeadv_T	Number of adverbs of time divided by the number of tokens
vocatives_S	Number of vocatives divided by the number of sentences
nptheme_S	Number of nominal elements in theme position divided by the number of sentences
numbertheme_S	Number of numbers in theme position divided by the number of sentences
pptheme_S	Number of prepositions in theme position divided by the number of sentences
advtheme_S	Number of adverbs in theme position divided by the number of sentences
texttheme_S	Number of conjunctions in theme position divided by the number of sentences
whtheme_S	Number of wh-elements in theme position divided by the number of sentences
disctheme_S	Number of discourse markers in theme position divided by the number of sentences
nonfinite_verbstheme_S	Number of infinitives with <i>zu</i> in theme position divided by the number of sentences
subordconjtheme_S	Number of subordinating conjunctions in theme position divided by the number of sentences
verbtheme_S	Number of verbs in theme position divided by the number of sentences
incompletesentences_S	Number of incomplete sentences divided by the number of sentences
cohesiveadverbs_T	Number of cohesive adverbs divided by the number of tokens
emoji_T	Number of emojis divided by the number of tokens
hashtag_T	Number of hashtags divided by the number of tokens
jodelwords_T	Number of instances of Jodel specific words (eg. <i>jodel, karma</i>) divided by the number of tokens
socialmediawords_T	Number of instances of social media specific words (eg. <i>upvote</i>) divided by the number of tokens
colloquialisms_T	Number of colloquialisms divided by the number of tokens
fm_T	Number of foreign language material divided by the number of tokens
emojitHEME_S	Number of emojis in theme position divided by the number of sentences
emojionly_S	Number of sentences consisting only of emojis divided by the number of sentences
hashtagtheme_S	Number of hashtags in theme position divided by the number of sentences
hashtagonly_S	Number of sentences consisting only of hashtags divided by the number of sentences

reftheme_S	Number of references to another user in theme position divided by the number of sentences
correctiontheme_S	Number of corrections in theme position divided by the number of sentences
personaldetails_P	Number of personal details (e.g. gender and age) divided by the number of sentences
modalpart_T	Number of modal particles divided by the number of tokens
focuspart_T	Number of focus particles divided by the number of tokens
multipart_T	Number of multi-word particles divided by the number of tokens
contrverbpron_T	Number of contractions with a verb and a pronoun (eg. <i>gehts, habs</i>) divided by the number of tokens
contrpronpron_T	Number of contractions with two pronouns (eg. <i>ers, sies</i>) divided by the number of tokens
contrkoupron_T	Number of contractions with a conjunction and a pronoun (eg. <i>weils, obs</i>) divided by the number of tokens
contrpreart_T	Number of contractions with a preposition and an article (eg. <i>beim, am</i>) divided by the number of tokens

Extraction and Classification of Speech, Thought, and Writing in German Narrative Texts

Luise Schricker and Manfred Stede

Applied Computational Linguistics

Dept. of Linguistics

University of Potsdam / Germany

schricker|stede@uni-potsdam.de

Peer Trilcke

Literary Studies

Dept. of German Studies

University of Potsdam / Germany

trilcke@uni-potsdam.de

Abstract

For various purposes of narrative text analysis, it is helpful to identify *speech and thought events*: material that is uttered or imagined by some protagonist. This task commonly distinguishes between direct and indirect speech, but we will also consider free indirect and reported speech here. Specifically, we build upon earlier work by Brunner (2015), who presented an annotated German corpus of narrative texts as well as an automatic analysis system. We propose a variety of extensions and are able to substantially improve on the original results for all four categories.

1 Introduction

Identifying speech, thought and writing (henceforth STWR) of characters is a central task in automatically understanding narrative text. It is commonly divided into several categories, the most widely-researched (on the computational side) being *direct speech*. In the following example from Fontane's *Effi Briest*, the speech content is marked by italics and the so-called 'inquit formula' by boldface:

(1) "Warum lacht ihr?", **sagte Effi** pikiert. "Was soll das heißen?"

("Why are you laughing?", **said Effi** slightly offended. "What is it supposed to mean?")

The next example, taken from the same novel, illustrates *indirect speech*, which is commonly expressed with dependent clauses (italics) and a framing clause (boldface):

(2) In diesem Augenblick trat Wilke vom Saal her ein und **meldete**, dass er alles schon nachgezählt und alles vollständig gefunden habe;

(At this point Wilke entered from the hall and **announced** that he has counted everything and found it to be complete;)

In *free indirect speech* the character speaks with the narrators voice, which we also illustrate with an example from *Effi Briest*, with Effi's thought process in italics:

(3) Sie hatte Mühe, sich zurechtzufinden. *Wo war sie? Richtig, in Kessin, im Hause des Landrats von Innstetten, und sie war seine Frau, Baronin Innstetten.*

(She had trouble with orientation. *Where was she? Right, in Kessin, in the house of Instetten, and she was his wife, Baroness Innstetten.*)

Finally, *reported speech* has the most distance to the actual words produced by the character, which the narrator may summarize, as in this example from the same source:

(4) Sie sprachen noch eine Weile so weiter, wobei sie sich ihrer gemeinschaftlichen Schulstunden und einer ganzen Reihe Holzapfelscher Unpassendheiten mit Empörung und Behagen erinnerten.
(They talked like that for another while, remembering with indignation and with pleasure the common school hours and a number of Holzapfel's inappropriate moments.)

More detailed descriptions of the STWR types can be found in narratological studies, such as Genette (1998), Jessing et al. (2007), or Martinez and Scheffel (2012).

From the computational viewpoint, attributing speech, thought and writing to the characters producing them is a fundamental step in following the storyline of a narrative, and the first central step is to be able to *identify* STWR material. This, in turn has two steps: making the distinction between direct/indirect/free ind./reported, and that between speech, thought and writing. For German, there has not been a lot of work on these tasks, and only one sizable annotated corpus was available at the time of our system's development.¹ It was com-

¹In the meantime, a bigger corpus with STWR annotations

piled by Brunner (2015), and her work constitutes the starting point of our study. Our main contribution is to propose an extended set of features used by an automatic classifier, as well as a few other improvements that substantially improve on Brunner’s original results and thus can be regarded as a new state of the art for this dataset.

The paper is structured as follows: After introducing the corpus and briefly reviewing earlier research in Section 2, we explain our experiments in Section 3, report and discuss the results in Section 4, and then conclude in Section 5.

2 Corpus and Related Work

2.1 Corpus

The corpus annotated by Brunner (2015) is made up of thirteen short German narratives. The texts were written between 1787 and 1913 and consist of overall roughly 57,000 tokens. They were annotated for the 12 categories that result from combining speech, thought and writing each with the four different types direct, indirect, free_indirect and reported. Several attributes, mostly signaling border cases of annotation, were also included.

The corpus has several strengths and weaknesses. It is not a balanced dataset, insofar as the number of instances for each class differ considerably. This is most pronounced in the case of *free_indirect* STWR which only amounts to 110 instances compared to 1038 instances for *direct* STWR. Furthermore, the whole corpus was annotated by a single person, Brunner herself, thus making the annotations susceptible to subjectivity.

On the other hand, the texts of the corpus were carefully chosen to represent a wide selection of narrative texts. Complete short texts were selected, rather than excerpts (with two exceptions). The texts have diverse dates of origin spanning over a 100 years, and authors of both genders are represented. Also, care was taken to ensure a mix of different narrative perspectives, first person and third person. The texts differ in the punctuation schema used as well as in orthography. We propose that these characteristics provide a realistic set-up for developing a system for practical use in literary studies.

has been beta-released by the *Redewiedergabe-Projekt* (Brunner et al., 2019). This newly released corpus could not be used in the present study though because of time constraints.

2.2 Earlier research

Most earlier work on speech event identification targets a different genre, viz. news text. For instance, Krestel et al. (2008) develop a rule-based system for extracting *direct* and *indirect* speech from newspaper articles. Their system consists of two components: a reporting verb marker and a reported speech finder, which uses six patterns combining a reporting verb, a speech source, and one or two clauses containing the speech content. The authors evaluate their system on seven newspaper articles from the Wall Street Journal corpus with a total of 6100 words and report a recall of 0.83 and a precision of 0.98 on this test set.

Sarmento and Nunes (2009) develop a system, which they call *verbatim*, for the extraction of *direct* and *indirect* quotes from Portuguese news texts. Their system uses 19 patterns and a list of 35 speech words to extract quotes along with the respective speakers and speech acts. The system is evaluated manually: of 570 extracted quotes, 68 are considered errors, yielding a precision of 0.88; recall is not reported.

Pareti et al. (2013) train and evaluate two machine learning (ML) approaches to extract *direct*, *indirect* and mixed quotes from two news corpora. In order to avoid compiling a list of common speech verbs, which cannot account for all verbs used as cues for quotations, the authors train a classifier to identify attribution verb cues. They experiment both with a token-based approach and with a system that classifies parse nodes. These experiments result in F1 measures of up to 0.60 for indirect and 0.94 for direct quotations with exact boundary matching.

Turning to narrative, Brunner (2015) (see also (Brunner, 2019)) implemented both a rule-based and an ML based system for the automatic annotation of the STWR categories in German narrative. The best F1 scores that were achieved with her systems range from 0.40 for *free indirect* STWR to 0.87 for *direct* STWR. For the types *reported* and *indirect* STWR the best F1 values are 0.58 resp. 0.71. These results were achieved with sentences as a basic unit: Brunner considers a sentence as belonging to a certain STWR class, if at least one STWR instance belonging to this class appears somewhere within it. If a sentence contains instances of several different STWR classes, it receives multiple labels. In addition, Brunner also experimented with segments of sentences, which

roughly correspond to clauses, but she discarded that approach because it lowered the performance of her system. Brunner presented her implementations as prototypes and suggested that further features be explored for potentially improving the systems. Our experiments reported below build directly on Brunner’s work.

3 Experiments

3.1 Approach

While Brunner’s work focuses on sentence-based classification, our work targets the automatic recognition of sub-sentential boundaries. Hence, we train our classifiers on the segments of sentences Brunner used for her “extra” experiments. In order to also allow the annotation of unseen texts, we reimplemented the tokenization algorithm described by Brunner and slightly extended it toward a broader quotation recognition method by adding heuristics for disambiguating apostrophes and quotation marks, following Percillier (2017).

3.2 Training and balancing

We follow Brunner in training a binary classifier for each STWR type, to allow multiple labels per segment, as STWR instances can naturally occur in nested form. After extracting STWR instances from the corpus per type, each training data set is transformed into feature representations. At this point the segments are still in the order as they appear in the texts, allowing us to also build sequence-based features (and we will specifically test their impact on the results). For the SVM classifier, the feature representations are also being scaled for efficiency reasons. The transformed data set is then split into stratified train and test sets. This step differs from Brunner, who did not set aside a dedicated test set, but rather evaluated the ML classifiers via cross validation (CV) on the whole training set. As she for the most part abstains from parameter tuning, the CV results are probably not biased.² For our study, on the other hand, we want to run parameter tuning in order to find the optimal configurations for the ML methods. Thus, 25% of the data for each class are set aside for testing. Due to this difference in evaluation data, some of

²She mentions experiments with different ML methods and ways to remove class imbalances, but does not give concrete results. Her reported results were all achieved by the same configuration: a Random Forest with 500 trees, which can each inspect eight features when adding a new node.

our results presented in Sct. 4 cannot be directly compared to Brunner’s results.

Experiments and parameter tuning were performed via stratified ten-fold cross validation on the training set. For feature scaling, the test-train split and cross validation, implementations provided by Scikit-learn 0.20.2 (Pedregosa et al., 2011) are used. In every fold the train set is further adjusted to tackle the class imbalance problem, which is especially pronounced in the case of the free indirect class. Brunner used the technique of *oversampling* on the training data to achieve an equal class distribution. For this technique instances are drawn randomly from the smaller class until an equal class distribution is achieved. While the training data can be adjusted to achieve equal class distribution, data set aside for evaluation or validation stays untouched to ensure reliable results.

Specifically, we use two alternative strategies to counter the class imbalance problem and compare them to the oversampling technique used by Brunner. The *Synthetic minority over-sampling technique (SMOTE)* (Chawla et al., 2002) uses a combination of undersampling the majority class and oversampling the minority class. Furthermore, the oversampled instances are synthetically created by adjusting features in the direction of one of the k nearest neighbors of the instance. For this, we use the imbalanced-learn package (Lemaître et al., 2017), version 0.4.3.

While SMOTE is domain agnostic, our second balancing strategy is a domain-specific *data augmentation* method. Here, features that were expected to be variable in naturally occurring data were manually selected for each class. As with oversampling, instances are drawn from the minority class until an even class distribution is achieved. For each drawn instance, a random subset of the augmentation features for this class are chosen and the instance’s respective feature values are changed: Boolean features are negated and real-valued features are substituted by a random value within the interval of possible values for this instance.

3.3 Classifiers

Our reporting type classifier consists of four separate binary classifiers that determine whether an instance contains at least one occurrence of the class *direct*, *indirect*, *free indirect* and *reported*, respectively. We do not use a multiclass approach because an instance can contain nested or sequentially

appearing STWRs that belong to different classes. Furthermore, the STWR annotations which have the *ambiguous* attribute belong to two classes.

We evaluate three different ML techniques on our task: Random Forest, Support Vector Machines (SVM) and Multi-Layer Perceptron (MLP). For all three ML techniques implementations provided by Scikit-learn 0.20.2 (Pedregosa et al., 2011) are used: *RandomForestClassifier*, *SVC* and *MLPClassifier*. Some parameters, for example the number and size of the hidden layers (MLP) or the minimal number of samples per leaf (Random Forest), were adjusted via grid search and cross validation on the training data. For the others, we largely adopted the default parameter options, but the MLP’s *early_stopping* parameter was set to True instead of False (default) and the *tolerance for the optimization (tol)* was set to 0.01 instead of 0.0001 (default). These settings achieve an early stop of the training process when the loss or validation score does not improve by at least *tol* for ten consecutive iterations. This is a regularization measure to prevent overfitting, which is important given the small number of training samples. The SVC’s *tol* parameter was also set to 0.01 for the same reasons.

Following the annotations performed by the reporting type classifier, a simple rule-based system adds a further layer of annotation: The detected STWR instances are categorized as either speech, thought or writing. The *direct* and *free indirect* classes are always annotated with their majority class. For the *indirect* and *reported* classes, we search the respective segment for occurrences of words in a *reporting word list* (see below). If exactly one such word is found in the segment, it is marked correspondingly as speech, thought or writing. In the absence of a reporting word, we use the majority class.

3.4 Features

We now describe our extensive feature set, which was developed on the basis of Brunner (2015) – henceforth: B15 – and was extended with features adapted from other literature or devised by us.

In order to keep track of sequential features, we use a backlog with information about the previous ten segments’ labels. We see this as potentially useful because certain STWR types tend to appear in blocks, for example when a conversation between characters is reported. B15 also reported this ob-

servation, but did not use corresponding features.

Our set-up of the data flow is such that only gold labels can be used for the sequential features instead of real predictions by the system, which can only be made after the system has been trained, i.e. after the dataset is already split into stratified train- and test-sets. Those sets lose segment order information, as they are constrained to contain the same distribution of classes. More sophisticated experiments with system-generated sequential features are left for future work. However, to ascertain their general influence, we evaluated our classifier both with and without the gold label features, thus providing upper and lower bounds for performance.

In the course of feature extraction the following tools and packages are used: *pandas* 0.21.0 (McKinney and others, 2010) for data handling, *numpy* 1.13.3 (Walt et al., 2011) for vector operations, *spaCy* 2.0.12 (Honnibal and Montani, 2019) for most linguistic processing tasks (e.g. part of speech (POS) tagging, and named entity recognition), *scipy* 1.0.0 (Jones et al., 2001) for computing cosine distances, *gensim* 3.1.0 (Řehůřek and Sojka, 2010) for the handling of word vectors and the *RFTagger*³ (Schmid and Laws, 2008) for morphological analysis. As spaCy’s German lemmatizer turned out to not work well, *GermanLemma* 0.1.1 (Konrad, 2019) was used as an alternative.

3.4.1 Token features

POS: distribution of POS tags, using both the *TIGER Treebank tags* and *Google Universal POS tag set*. B15 used the STTS tagset and a self-made broader tagset that combined some related tags of the STTS .

Named Entities: Presence of NEs and their types (person, location, organization, misc). Similar features were also used by Fernandes et al. (2011) and Pariet et al. (2013).

Special tokens: Presence of colons (also used by B15), question marks (Cohn (1978) mentions interrogations as one of the characteristic traits of *free indirect* thought) and quotation marks (generally used in the literature for finding *direct* speech).⁴ Like B15, we also use segment-end fea-

³<http://www.cis.uni-muenchen.de/~schmid/tools/RFTagger/>

⁴We use the number of opening and closing quotation marks within the segment; whether the segment is in quotes;

tures: comma (as indicator of a following inquit formula for direct STWR), emphatic punctuation marks (? ! - –), and special combinations associated with the *direct* class.

3.4.2 Morphological features

Person: The change from third to first/second person is often considered an indicator of a change from narration to characters' words (Cohn, 1978). We use the frequency of first and second person pronouns in the segment as separate features (whereas B15 and Mamede and Chaleira (2004) put both pronoun types into the same category); frequency of third person pronouns (cf. B15); Boolean features indicating whether only third person, only first person or a mixture of first and third person was used in the previous five segments. These features give a broader picture of the usage of person in the text.

Mode: Partly adapted from B15, we use Boolean features indicating whether any verb and whether all verbs in the segment are in indicative or in subjunctive mode.⁵

Tense: The basic tense of narration is often past tense while characters' words mostly use present tense (Cohn, 1978). *Free indirect* STWR is an exception, as it displays features of the narrator's voice, e.g. by using the same tense (Jessing et al., 2007; Martinez and Scheffel, 2012). We use Boolean features indicating whether any verb and whether all verbs in the segment are in a form of present tense and past tense, respectively.

3.4.3 Grammatical features

This group comprises a set of Boolean features indicating whether

(i) the previous segment ended with a comma and this segment contains a verb, suggesting an embedded sentence. Embedded sentences, according to B15, are frequently part of *indirect* STWR, especially if they contain verbs in subjunctive mode.

(ii) the segment contains a form of the verb *würden* in combination with an infinitive verb or by itself. Fabricius-Hansen (2002) observes the combination of *würde* ('would') with an infinitive verb as a possible indicator of *free indirect* STWR. B15 also adapted this observation in her system.

number of contiguous previous segments in quotes (meant to prevent errors caused by missing closing quotation marks).

⁵Martinez and Scheffel (2012) mention the verb form of first person indicative with present tense as one of the typical features of *direct* speech.

(iii) any reporting word within the segment has a noun or prepositional complement, which B15 treated as an indicator of *reported* STWR. (Reporting words are further described below.)

(iv) any reporting word within the segment has a sentence or infinitive complement, which B15 saw as an indicator of *indirect* STWR.

3.4.4 Candidate speaker features

Candidate speakers are usually extracted with the aim to attribute quotes to their speakers, e.g. (Elson and McKeown, 2010; Mamede and Chaleira, 2004). Here, the idea is that the appearance of a candidate speaker might also be useful for detecting STWR. We regard pronouns, named entities of the person type and head nouns that belong to the *Person* category as possible speakers. The list of nouns that belong to the *Person* category was gathered by recursively extracting the hyponyms of all synsets of the word *Person* from GermaNet (Henrich and Hinrichs, 2010). The process for extracting candidate speakers is adapted from Elson and McKeown (2010) and Jannidis (2017).

Our feature set consists of a Boolean feature indicating whether the segment's subject is a candidate speaker; the number of candidate speakers in the segment; and the candidate speaker features of the previous segment.

3.4.5 Reporting word features

We used the list of reporting words that B15 compiled by combining three sources: a linguistically motivated list of reporting verbs, words from a corpus that were extracted via pattern matching, and related words (verbs and nouns) mined from a thesaurus. Each word was given a penalty value representing the degree of "typicality" for a reporting word. If a word can refer to exactly one type of reporting act (speech, thought or writing), this type is recorded in the list — examples are *sagen* ('say') for the speech category and *denken* ('think') for the thought category. For some words, a marker is set to indicate that the word can not be used as a framing clause of *direct* or *indirect* STWR but only for the reporting type. Examples of such words are *plaudern* ('chat') and *befragen* ('interrogate').

Reporting word list features (i) For each penalty value, Boolean features indicate whether at least one reporting word of this value or of lower value appears in the segment. (Adapted from B15.)

(ii) Boolean features indicating whether the segment contains any reporting word that is a verb and

whether it contains any that is a noun.

(iii) For each penalty value, Boolean features indicate whether at least one reporting word with lower or equal penalty value with a *reporting marker* is contained within the segment. (B15 distinguishes these reporting words in her rule-based approach, but not in the ML implementation.)

(iv) For each penalty value, the number of reporting words of this (or a lower) value that are contained in the segment.

(v) Numbers of reporting verbs and reporting nouns contained in the segment.

(vi) For each penalty value the number of reporting words of this (or a lower) value with a *reporting marker* within the segment.

(vii) The reporting word features of the previous segment.⁶

Word vector features: Lists of reporting words are often used for quotation extraction, see e.g. (Krestel et al., 2008; Clergerie et al., 2009; Sarmento and Nunes, 2009; Brunner, 2015), but their inherent inflexibility poses a problem. STWR instances can be framed very differently. Especially literary texts can convey reporting acts with potentially infinite variance. Paret et al. (2013) try to circumvent this problem by implementing a separate classifier for detecting reporting words. Glass and Bangay (2007) use a list of features to determine whether a verb is a reporting verb. Here, we implemented a different approach to the problem.

In addition to looking up words in the reporting word list, similarity values of word vectors were used to achieve a more general indication of the appearance of reporting words. To this end, prototypical word vectors for reporting words and for reporting words with the *reporting marker* were computed. This was done by averaging all word vectors representing the words in the reporting word list with penalty 0 (i.e. the most typical words) and those with penalty 0 and a *reporting marker*. These average vectors can be considered as prototypical representations of the general reporting word group. All lemmata of the words contained in a segment were compared in turn to the two prototypical word vectors using the cosine similarity measure. The maximum similarities to each of the vectors were

⁶As B15 stated in her description of the segment-based ML experiments, instances of *indirect* reporting are almost always split up into two segments: the framing clause and the dependent clause. Adding the reporting word features of the previous segment should reestablish the connection between the two segments.

then added as features. This way the appearance of a word which is not contained in the reporting word list, but which is nonetheless sometimes used as a reporting word, can be detected.

Two word vector models were compared to each other in the experiments: a model trained on the KOLIMO corpus⁷ (Herrmann and Lauer, 2017) and a model trained on the German Wikipedia and a corpus of German news articles. The Wikipedia model is available as a pretrained model⁸ (Müller, 2018). Its suitability for the present task is not clear because of the genre difference to our task (contemporary non-narrative texts versus narrative written between 1787 and 1913). The data of the KOLIMO corpus is in principle better suited for the task because it contains German narrative with a focus on Modernism, a period which roughly spans the years 1880 to 1930. KOLIMO is a broad corpus though, comprising also many texts from earlier periods, as well as non-narrative. Furthermore, the KOLIMO distribution is a beta release, which still contains some noise and does not have a consistent format (Herrmann and Lauer, 2017). Therefore, both models have their merits and disadvantages.

For preprocessing and training of the KOLIMO word vectors, the toolkit provided by Müller (2018) was used. This toolkit builds on Google's word2vec tool as described by Mikolov et al. (2013).

3.4.6 Other word features

(i) The percentage of **deictic words** in the segment. The list of deictic expressions was taken from B15: *heute, morgen, gestern, jetzt, hier* (today, tomorrow, yesterday, now, here). Deictic words can indicate the speaker's perspective, i.e. whether the speaker is likely a character, who is situated in the narrative's here and now, or whether the speaker is a narrator whose frame of reference can be different from the intratextual one (Cohn, 1978).

(ii) A Boolean feature indicating whether the segment begins with a **conjunction** that can indicate the *indirect* class. The list is taken from B15: *dass* ('that'), *ob* ('whether'), *wo* ('where'), etc.

(iii) The percentage of **modal particles** in the segment, whose appearance can be an indicator of character speech. The list of modal particles was also taken from B15: *ja, nein, wohl, schon, eigentlich, sowieso, eben* ('yes', 'no', 'already',

⁷<https://kolimo.uni-goettingen.de>

⁸<https://devmount.github.io/GermanWordEmbeddings/>

‘anyway’ and various particles with no English equivalent).

(iv) The percentage of words within the segment that are associated with **negation**. Our list contains *nein* (‘no’), *nicht* (‘not’), *kein* (‘no’). This feature was added because B15 observed in her description of the STWR corpus that the classes *indirect* and *reported* often display markers of negation and non-factuality. She did not use respective features in her own implementations.

(v) Boolean features indicating the appearance of words describing **facial expressions, gestures and voice**. These features were inspired by Tencini (2010)’s narratological study on the function and relevance of coverbal language in literature. The reasoning is that the appearance of words indicating coverbal language might be an alternative way to recognize the context of an STWR instance in the absence of reporting words. Word lists were manually crafted for each category:

- facial expressions: *Gesicht* (‘face’), *Mund* (‘mouth’), *Augenbraue* (‘brow’), *Auge* (‘eye’), *Stirn* (‘forehead’), *Lippe* (‘lip’), *Nase* (‘nose’), *Nasenflügel* (‘side of nose’)
- gestures: *Hand* (‘hand’), *Arm* (‘arm’), *Handfläche* (‘palm’), *Finger* (‘finger’), *Schulter* (‘shoulder’), *Faust* (‘fist’)
- voice: *Stimme* (‘voice’), *Ton* (‘sound’), *Tonhöhe* (‘pitch’), *Tonfall* (‘tone’), *Stimmlage* (‘register’), *Atem* (‘breath’)

(vi) A Boolean feature indicating whether any word in the segment is repeated. According to Cohn (1978) the **repetition** of words can be a feature of characters’ language.

3.4.7 Sequential label features

For each reporting type, we compute (i) Boolean features that indicate whether the previous segment and whether any of the previous five segments was labeled with this type, and (ii) the number of segments among the previous ten segments which were labeled with the respective STWR type (this should help to detect blocks of STWR). Our final feature in this group is the overall number of STWR instances of all reporting types annotated within the previous ten segments.

3.4.8 Length/position features

In the last group, we use these features: (i) Token- and character-based lengths of the current and the

previous segment, as an approximation of the style of the segment by indicating whether longer or shorter words are used. (B15 only used the token-based sentence/segment length.)

(ii) The sum of the token-based lengths of the current and the previous segments and the sum of the character-based lengths.

(iii) Boolean features indicating whether the current or the previous segment constitutes the end of a paragraph (taken from B15).

4 Results and discussion

Below we report our results on overall performance, in comparison to the segment-based results of B15 as far as possible. We will explore the difference between using sequential label features and ignoring them, and between the two word vector models. Regarding the various *balancing* techniques, we found that oversampling and our own data augmentation method outperformed the SMOTE method. As for the different ML approaches, Random Forest yielded the best performance for the overall best configurations (BEST-ALL). These were evaluated on the test set, results are shown in Table 2. Note that the BEST-ALL configurations are not always the same as the best configurations within each individual class (BEST-IND), which were derived by CV on the training data, and whose results are listed in Table 1, sometimes stemming from other classifiers than Random Forest (for reasons of space, we do not provide the detailed comparison). The BEST-ALL configurations were picked by adding the F1 values for each of the four classes and choosing the combination which achieves the highest sum.

The BEST-ALL configuration with sequential label features uses the Random Forest model with KOLIMO word vectors and oversampling. Both KOLIMO vectors and oversampling were chosen as frozen parameters for the ML parameter adjustment phase, which may have influenced the result. For the *indirect* and *reported* classes considered individually (BEST-IND), the Random Forest model with KOLIMO word vectors and data augmentation slightly outperforms the BEST-ALL configuration, indicating untapped potential in data augmentation.

Without sequential label features the BEST-ALL configuration is the Random Forest model with Wikipedia word vectors and oversampling. Three out of four BEST-IND models also use the Wikipedia word vectors. This result is somewhat

surprising, as noted earlier, and may indicate that word vector models trained on non-narrative data can be used for work on narrative texts.

For all four classes, the BEST-IND F1 scores achieved with CV and gold sequential label features outperform the models without sequential label features. The *free indirect* class profits most from the sequential label features, with an increase of 52.17 points F1 score.⁹

The baseline of Brunner's system was only compared to the results of the BEST-IND models without sequential features, derived via CV, in order to ensure a fair comparison. The system developed in this study outperforms Brunner's for every class,¹⁰ with improvements of 17.02 points F1 score for the *indirect* class and 15.81 points for the *direct* class. The *reported* and *free indirect* classes show the least improvement with an additional 9.70 points resp. 11.39 points F1 score.

The BEST-ALL configurations with and without sequential label features were evaluated on the test set. A simple baseline was also evaluated to provide an additional point of comparison. This baseline uses the same configuration as the STWR classifiers, but it only has one feature, viz. the segment's character-based length. Results on the test set are listed in Table 2. The results of the simple baseline are considerably lower than the ML model's results, both with and without sequential label features. For all STWR types except the *free indirect* class the results achieved on the test set are within a range of two points on either side of those achieved with CV on the training set. Results that can be reproduced on the test set are considered reliable. For the *free indirect* class, the F1 scores on the test set, with and without sequential label features, are more than 10 points lower than those achieved via CV. This is likely a consequence of overfitting to the training set, which is difficult to avoid for such a small dataset.

Overall, the results of the models with sequential label features are promising, with an F1 score of

⁹This substantial increase could be attributed to the fact that most of the *free indirect* samples in the STWR corpus can be found in one text, *Der Irr* by Georg Heym. Therefore, if a sample's sequential label features contain instances of *free indirect*, there is a high probability that the sample belongs to this particular text.

¹⁰The BEST-IND results were used as the point of comparison to Brunner's baseline, but the BEST-ALL configuration also improves upon Brunner's baseline in every class. Brunner's sentence-based results are not reported here, as they cannot be compared to the segment-based results of the present system.

95.01 for the *direct* class, 79.48 for the *indirect* and 70.13 for the *free indirect* class. Only the *reported* class, with a score of 49.28, is still not recognized well. The difference of the scores achieved with and without sequential label features, i.e. between the upper and lower bounds of evaluation, are similar to those observed for CV. *Free indirect* is the class that suffers the most severe decline in accuracy with the loss of the sequential label features, i.e. 58.37 points F1 score, compared to a loss of 52.17 points F1 score for CV. This dependency on the gold labels shows that the class by itself is not well recognized by the classifier. The problem might be alleviated by using a bigger training data set. The *reported* class on the other hand is the class that loses least accuracy when the sequential label features are removed, indicating that *reported* STWR does not often occur blockwise within the STWR corpus.

Finally, we evaluated the second classifier (for distinguishing speech, thought and writing) on the test set. Results are listed in Table 3. Note that only positively classified instances (real predictions, not gold labels) are further annotated with speech, thought and writing. The classification scheme based on the majority class works well for the classes *direct* and *free indirect*. The *direct* class is classified correctly as speech with an F1 score of over 90 points. Thought and writing are minority classes with counts of up to eleven instances, and thus can be ignored. The *free indirect* class only consists of instances that are correctly labeled with the majority class thought, the F1 score reaches 90 points with sequential label features and 60 points without. This shows the influence of the first classifier's performance, because instances that do not belong to a class but are incorrectly recognized as such, are also incorrectly classified with one of the types speech, thought or writing. The results for the *indirect* and *reported* classes are mixed. These types are more heterogeneous than the other two, i.e. their majority class covers a smaller percentage of the instances. Before defaulting to the majority class, the classifier attempts to classify *reported* and *indirect* instances by searching for reporting words. The F1 scores for the classes with more than zero members range between 28.57 and 69.46. This shows that the classification methods do not completely fail in the case of minority classes but that there is still room for improvement.

Model	STWR	Seq	Prec.	Recall	F1
Brunner	direct	-	71.00	66.00	69.00
STWR	direct	-	83.25	86.58	84.81
STWR	direct	+	96.47	95.19	95.80
Brunner	indirect	-	40.00	38.00	39.00
STWR	indirect	-	65.46	49.12	56.02
STWR	indirect	+	84.50	74.80	79.21
Brunner	free ind	-	28.00	15.00	20.00
STWR	free ind	-	23.79	47.15	31.39
STWR	free ind	+	87.79	80.46	83.56
Brunner	reported	-	39.00	31.00	34.00
STWR	reported	-	39.61	49.63	43.70
STWR	reported	+	42.56	62.93	50.51

Table 1: Results (BEST-IND) of the parameter tuning process via cross validation compared to Brunner’s segment-based results.

Model	STWR	Seq	Prec.	Recall	F1
Baseline	direct	-	28.39	59.21	38.37
STWR	direct	-	87.09	83.26	85.13
STWR	direct	+	96.54	93.51	95.01
Baseline	indirect	-	10.85	43.84	17.40
STWR	indirect	-	61.82	50.25	55.43
STWR	indirect	+	84.07	75.37	79.48
Baseline	free ind	-	03.27	72.73	6.26
STWR	free ind	-	42.86	06.82	11.76
STWR	free ind	+	81.82	61.36	70.13
Baseline	reported	-	12.00	39.56	18.41
STWR	reported	-	41.10	49.45	44.89
STWR	reported	+	43.64	56.59	49.28

Table 2: Results (BEST-ALL) of the evaluation of the STWR classifiers on the test set.

5 Conclusion

We were able to substantially improve the original classification results on a German corpus annotated for speech, thought and writing (Brunner, 2015). This is largely due to changes in the feature set, which we described in detail. In addition, we tested techniques for handling class imbalance: oversampling, SMOTE and our own (domain-specific) data augmentation method. Also, we demonstrated the utility of sequence based features (experiments with predicted values are left for future work, though), and we compared the contributions of two different word vector models.

References

- Annelen Brunner, Lukas Weimer, Ngoc Duyen Tanja Tu, Stefan Engelberg, and Fotis Jannidis. 2019. Das Redewiedergabe-Korpus. Eine neue Ressource. In Patrick Sahle, editor, *Digital Humanities: multimodal multimodal*. 6. Tagung des Verbands Digital Humanities im deutschsprachigen Raum e.V. (DHd 2019), pages 103–106.
- Annelen Brunner. 2015. *Automatische Erkennung von*
- Cathrine Fabricius-Hansen. 2002. Nicht-direktes Referat im Deutschen – Typologie und Abgrenzungsprobleme. In Cathrine Fabricius-Hansen, Oddleif Leirbukt, and Ole Letnes, editors, *Modus, Modalverben, Modalpartikeln*, volume 25 of *Linguistisch-philologische Studien*, pages 6–29. Wissenschaftlicher Verlag Trier, Trier.
- William Paulo Ducca Fernandes, Eduardo Motta, and Ruy Luiz Miliú. 2011. Quotation extraction for

STW	STWR	Seq	Co	Prec.	Rec.	F1
S	direct	-	389	85.12	100.0	91.96
T	direct	-	9	0.00	0.00	0.00
W	direct	-	1	0.00	0.00	0.00
S	direct	+	436	94.17	100.0	96.70
T	direct	+	11	0.00	0.00	0.00
W	direct	+	1	0.00	0.00	0.00
S	indirect	-	42	31.43	26.19	28.57
T	indirect	-	60	42.31	91.67	57.89
S	indirect	+	62	41.18	22.58	29.17
T	indirect	+	91	56.08	91.21	69.46
T	free ind	-	3	42.86	100.0	60.00
T	free ind	+	27	81.82	100.0	90.00
S	rep	-	66	39.02	96.97	55.65
T	rep	-	23	28.85	65.22	40.00
W	rep	-	3	33.33	33.33	33.33
S	rep	+	67	34.74	98.51	51.36
T	rep	+	33	31.82	42.42	36.36
W	rep	+	5	50.00	20.00	28.57

Table 3: Results of the evaluation of the speech (S), thought (T) and writing (W) classifier on the test set. The count column (Co) represents those instances identified by the STWR classifier, which are further annotated with the classes speech, thought and writing.

Redewiedergabe: ein Beitrag zur quantitativen Narratologie. Walter de Gruyter.

Annelen Brunner. 2019. Redewiedergabe–Schritte zur automatischen Erkennung. *Zeitschrift für germanistische Linguistik*, 47(1):216–248.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Éric de la Clergerie, Benoît Sagot, Rosa Stern, Pascal Denis, Gaëlle Recourcé, and Victor Mignot. 2009. Extracting and visualizing quotations from news wires. In *Language and Technology Conference*, pages 522–532. Springer.

Dorrit Cohn. 1978. *Transparent minds: Narrative modes for presenting consciousness in fiction*. Princeton University Press.

David K Elson and Kathleen R McKeown. 2010. Automatic attribution of quoted speech in literary narrative. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Cathrine Fabricius-Hansen. 2002. Nicht-direktes Referat im Deutschen – Typologie und Abgrenzungsprobleme. In Cathrine Fabricius-Hansen, Oddleif Leirbukt, and Ole Letnes, editors, *Modus, Modalverben, Modalpartikeln*, volume 25 of *Linguistisch-philologische Studien*, pages 6–29. Wissenschaftlicher Verlag Trier, Trier.

William Paulo Ducca Fernandes, Eduardo Motta, and Ruy Luiz Miliú. 2011. Quotation extraction for

- portuguese. In *Proceedings of the 8th Brazilian Symposium in Information and Human Language Technology*.
- Gérard Genette. 1998. *Die Erzählung*. Wilhelm Fink Verlag, 2 edition.
- Kevin Glass and Shaun Bangay. 2007. A naive salience-based method for speaker identification in fiction books. In *Proceedings of the 18th Annual Symposium of the Pattern Recognition Association of South Africa (PRASA '07)*, pages 1–6.
- Verena Henrich and Erhard Hinrichs. 2010. Gernedit - the germanet editing tool. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC 2010)*, pages 2228–2235, Valletta, Malta.
- Berenike Herrmann and Gerhard Lauer. 2017. Kolimo. a corpus of literary modernism for comparative analysis. <https://kolimo.uni-goettingen.de/about.html> (2019-3-10).
- Matthew Honnibal and Ines Montani. 2019. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.
- Fotis Jannidis. 2017. Netzwerke. In Fotis Jannidis, Hubertus Kohle, and Malte Rehbein, editors, *Digital Humanities*, pages 147–161. Springer.
- Benedikt Jessing, Ralph Koehnen, and Horst Weber. 2007. *Einführung in die Neuere deutsche Literaturwissenschaft*. Springer.
- Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001. SciPy: Open source scientific tools for Python. <http://www.scipy.org/>.
- Markus Konrad. 2019. Germalemma. <https://github.com/WZBSocialScienceCenter/germalemma>.
- Ralf Krestel, Sabine Bergler, and René Witte. 2008. Minding the source: Automatic tagging of reported speech in newspaper articles. *Reporter*, 1(5):4.
- Guillaume Lemaître, Fernando Nogueira, and Christos K Aridas. 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.
- Nuno Mamede and Pedro Chaleira. 2004. Character identification in children stories. In *International Conference on Natural Language Processing*, pages 82–90, Berlin, Heidelberg. Springer.
- M. Martinez and M. Scheffel. 2012. *Einführung in die Erzähltheorie*. C.H.Beck, 9 edition.
- Wes McKinney et al. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. SciPy Austin, TX.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*.
- Andreas Müller. 2018. GermanWordEmbeddings. <https://devmount.github.io/GermanWordEmbeddings/> (2018-06-20).
- Silvia Paretí, Tim O’Keefe, Ioannis Konstas, James R Curran, and Irena Koprinska. 2013. Automatically detecting and attributing indirect quotations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 989–999.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Michael Percillier. 2017. Creating and analyzing literary corpora. In Shalin Hai-Jew, editor, *Data Analytics in Digital Humanities*, pages 91–118. 05.
- Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA.
- Luis Sarmento and Sérgio Nunes. 2009. Automatic extraction of quotes and topics from news feeds. In *DSIE'09-4th Doctoral Symposium on Informatics Engineering*.
- Helmut Schmid and Florian Laws. 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained pos tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 777–784, Manchester, Great Britain. Association for Computational Linguistics.
- Maria Paola Tenchini. 2010. Reporting gesture and voice in reporting speech: Co-verbal language in literature. *CI-DIT*, 2010(2).
- Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. 2011. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

Argumentative Relation Classification as Plausibility Ranking

Juri Opitz

Leibniz ScienceCampus “Empirical Linguistics and Computational Language Modeling”
 Department for Computational Linguistics
 Heidelberg University
 69120 Heidelberg
 opitz@ccl.uni-heidelberg.de

Abstract

We formulate argumentative relation classification (support vs. attack) as a text-plausibility ranking task. To this aim, we propose a simple reconstruction trick which enables us to build minimal pairs of plausible and implausible texts by simulating natural contexts in which two argumentative units are likely or unlikely to appear. We show that this method is competitive with previous work albeit it is considerably simpler. In a recently introduced content-based version of the task, where contextual discourse clues are hidden, the approach offers a performance increase of more than 10% macro F1. With respect to the scarce *attack*-class, the method achieves a large increase in precision while the incurred loss in recall is small or even nonexistent.

1 Introduction

Argumentative relation classification (ARC) is dedicated to determining the class of the relation which may hold between two arguments or elementary argumentative units, EAUs¹. For instance, consider the following *premises* given the *topic* or *conclusion* (0) “Overall, marijuana is detrimental to your health.”:

- (1) *Use of marijuana causes chronic bronchitis and airflow obstruction.*
- (2) *Cannabis does not need to be smoked to receive its potential health benefits.*

In this case, (1) has a *positive stance* towards the conclusion (0); in contrast to (2), which has a negative stance towards the conclusion. Additionally, but not less importantly, we can say that (2)

¹Here, we use the term *elementary argumentative units* to denote clauses or small clause-complexes – e.g., (0), (1) or (2) – which can be ‘instantiated’ in an argumentative debate.

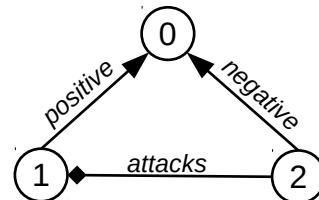


Figure 1: A small argumentation graph containing two general types of relations: premise-topic relations (class: negative/positive) and premise-premise relations (class: supports/attacks).

weakens (1) – it casts doubt about its generality by hinting at cannabis application methods which do not involve combustion or inhalation. In this work, we summarize all relations which aim at undermining or weakening another argument or premise (‘undercut’, ‘rebuttal’, etc.) as *attack*.² The EAUs from our example and their connecting relations are outlined in the graph in Figure 1.

In a rhetorically structured argumentative text³, (1) and (2) may appear in configurations such as *On the one hand (1), on the other (2); (1), however, (2)*, etc. Under these circumstances, discourse *context* can predict argumentative relations very well. However, when moving from such ‘closed scenario’ to a more ‘open-world setting’, e.g., where EAUs have been mined from heterogeneous documents, we need to determine relations based on their *content*. In this paper, we show that our method works well in both scenarios. In fact, it is in the more general and more difficult content-based setting, where our method provides the most benefits over previous work.

Systems which have learned to predict general argumentative relations have a decisive advantage when compared to systems that have ‘only’ learned to predict argumentative stances: in an argumenta-

²For a more ‘in-depth’ view and discussion of argumentative relations we refer the reader to, e.g., Pollock (1995), Walton (2009) and Besnard and Hunter (2014).

³E.g., an argumentative essay.

tive debate, often a debater does not choose to bring forth any argument which supports their stance on the topic. Instead, or additionally, they may choose to select an argument which also attacks the opponent’s most recent argument. Therefore, we need not only knowledge about the stances of arguments towards topics, but also about relations to other arguments. Our experiments show that our approach is a step towards this goal.

The remainder of this paper is structured as follows: After discussing related work in Section 2, we propose a simple reconstruction trick which allows us to embed an argumentative source-target pair in a relational discourse context yielding a plausible and implausible text variant (Section 3). In Section 4, we conduct experiments and ablation studies using (i) a standard task setup, where systems are allowed to see EAUs in their document context and (ii) a more difficult ‘content-focused’ task setup where systems are only allowed to see the spans of the EAU clauses. The code for this paper is available at <https://gitlab.cl.uni-heidelberg.de/opitz/pr4arc>

2 Related work

In this section, we first provide an overview of the data, and the data issues people are confronted with when developing argumentative relation classification (ARC) systems. We proceed with an overview of existing ARC approaches and conclude by touching on other related tasks.

Argumentative relation data For general argumentative relations, not many data sets have been developed. One of the largest data sets consists of 402 argumentative student essays and is henceforth denoted by ESSAY (Stab and Gurevych, 2014; Stab and Gurevych, 2017). It has been annotated, i.a., with EAU clauses and more than 3,000 relations which hold among them. By ESSAY-CONTENT, we denote a version of ESSAY from which discourse context is stripped and systems can only access the spans of EAU clauses (Opitz and Frank, 2019b). This setup is more difficult since systems have to learn to model the *content* of two EAUs in order to successfully predict their relation. ESSAY and ESSAY-CONTENT will be more extensively described in Section 4.1, where we also show that our method is efficient across both setups.

Another data set which is annotated with in-depth argumentative annotations is the Microtext corpus covering a variety of political debates in

Germany (Peldszus and Stede, 2016). While it has been annotated with a more fine-grained set of relations (e.g., *rebutting attack*, *undercutting attack*, *linked support*, *example support*) it is rather small in size (the recently extended version (Skeppstedt et al., 2018) contains about 700 relation tuples). Similar to ESSAY-CONTENT, a variant of the Microtext corpus exists where argumentative units are detached from discourse context (Wachsmuth et al., 2018). We believe that systems that have learned to predict argumentative relations based on the *content* of argumentative units have advantages over systems which focus too much on contextual discourse clues. For example, content-focused systems can better be expected to solve large-scale cross-document tasks where EAUs are mined from many heterogeneous documents. Our reconstruction trick provides one step towards this goal: it exploits *potential* discourse configurations without depending on seeing the *true* discourse context.

A key reason for the data scarcity of annotated general argumentative relations is that creating high-quality data for ‘premise-premise’ relations is a challenging task. Perhaps, it is more challenging than creating data for argumentative stance detection since topics or conclusions are often ‘a-priori’ well understood (e.g., *Cannabis should be legalized*) and always occur as the stance-relation target. In that sense, it may be easier and quicker to tell if an argument supports a conclusion compared to deciding whether an argument supports another argument.

ARC systems A linear SVM classifier that is trained on a diverse set of features provides competitive performance on ESSAY (Stab and Gurevych, 2017). A subsequent joint global graph optimization step, similarly to (Peldszus and Stede, 2015; Hou and Jochim, 2017), yields no further improvement for classifying the relations in this data. The SVM classifier incorporates features extracted from the EAU spans as well as their context (e.g., leading or trailing words). On ESSAY-CONTENT, where systems only see the EAU clause spans, the performance of the SVM suffers a loss of more than 10 pp. macro F1 (Opitz and Frank, 2019b) – an analysis indicates that the SVM focuses immoderately on features extracted from the EAU context and tends to neglect their actual content. This underpins the need for argumentative relation classification systems with deeper understanding of argumentation, i.e., systems that base their prediction on the actual

content of two EAUs – the method we present in this paper aims at this.

The first neural approach for ARC (Cocarascu and Toni, 2017) proposes a neural network with a Siamese structure (Koch et al., 2015; Mueller and Thyagarajan, 2016; Cocarascu and Toni, 2017). By means of a shared weight space it projects source and target EAU to a joint distributional vector space. Finally, it classifies the vector offset using a softmax-function. The authors conduct experiments on a data set which comprises texts about movies, technology and politics.

A similar model has been adopted recently where (symbolic) knowledge from large background knowledge-graphs is injected into the Siamese model by concatenating highly abstracted multi-hop knowledge paths to the source-target offset (Kobbe et al., 2019). Although there are consistent gains observed by including the knowledge, the gains appear to be relatively small. In this aspect, we believe that incorporating knowledge of the right form could make it possible to further enhance the system we propose in this paper. However, as of now, it is an active topic of discussion whether (symbolic) background knowledge may help in automatic argumentation and, even more so, which (form of) knowledge would be needed.

Computational argument mining and analysis
 Argumentation is ubiquitous and argumentative structures can be recovered from a broad spectrum of texts. For example, they can be recovered from online dialogue (Swanson et al., 2015; Budzynska et al., 2014) and scientific research articles (Lauscher et al., 2018a; Lauscher et al., 2018b), where, e.g., researchers may directly or indirectly convey arguments for why some method is better than another. By now, there exists a substantial body of research publications covering a variety of argument analysis topics. For a general overview, we refer the reader to Lippi and Torroni (2016) and Peldszus and Stede (2013).

Plausibility ranking Another task that can be addressed as a text plausibility ranking task is the resolution of difficult pronouns in the *Winograd Schema Challenge* (Levesque et al., 2012; Opitz and Frank, 2018). To resolve shell nouns and abstract anaphora (e.g., ‘I like that’.) Marasović et al. (2017) utilize syntactic patterns to gather plausible candidate resolutions from a background corpus in order to extend the scarce training data.

3 Context reconstruction and model

In this section, we first propose a simple reconstruction trick which allows us to build minimal pairs of plausible and implausible argumentative texts. Then, we describe a Siamese neural sequence ranking model which addresses the task of ranking texts according to their plausibility.

Constructing plausible and implausible argumentative discourse contexts Consider two EAU clauses a_2 (source) and a_1 (target) where we need to decide whether a_2 supports a_1 or a_2 attacks a_1 . In the absence of contextual discourse clues⁴, a system must learn to predict this relation by considering the semantic content of a_1 and a_2 . We approach this task by offering two alternative context reconstructions and asking our model in what context a_1 and a_2 are more likely to appear. More precisely, our reconstruction trick is as follows:

- (a) a_1 . Additionally, a_2 .
- (b) a_1 . Admittedly, a_2 .

where (a) signals that two argumentative units likely stand in a *support*-relation and (b) signals the opposite (‘attack’). In our experiments (Section 4), we also examine other possible discourse connectors for our reconstruction (e.g., moreover/however). From here, we ask our model which of the two reconstructions leads to a more plausible ‘reading’: (a) or (b)? E.g., consider the *cannabis*-example from Section 1; applying our reconstruction trick yields the following *implausible-plausible* minimal pair (r^- , r^+):

- (3a) [r_- Use of marijuana causes chronic bronchitis and airflow obstruction. **Additionally**, cannabis does not need to be smoked to receive its potential health benefits.]
- (3b) [r_+ Use of marijuana causes chronic bronchitis and airflow obstruction. **Admittedly**, cannabis does not need to be smoked to receive its potential health benefits.]

Clearly, (3b) constitutes a more plausible reconstruction compared with (3a). Exactly this is what we desire our model to learn: assessing the fine-grained differences between two texts which differ

⁴To name just one situation: consider a cross-document relation classification setup where a_1 stems from a different document than a_2 . Any specific textual discourse context would not only be more or less unimportant, but also bears the potential to confuse the system.

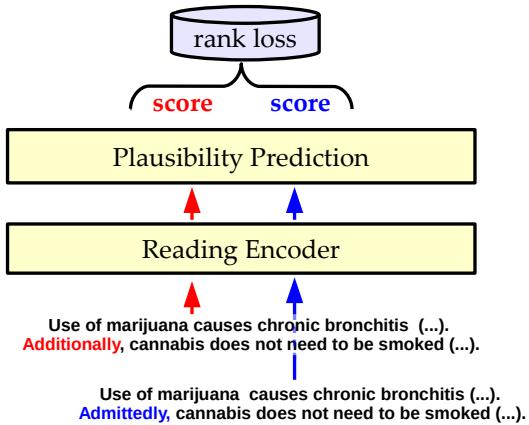


Figure 2: Siamese model outline. Two competing reading-reconstructions are fed through a Siamese encoder (Reading Encoder). The vectors are then mapped by means of a (Siamese) linear combination and a selu-activation onto two corresponding plausibility scores (Plausibility Prediction). By reducing the ranking loss, we force the model to assign higher scores to more plausible readings.

in only one phrase. This phrase, however, determines whether the text in its entirety is *implausible* or *plausible*.

3.1 Loss and model

Ranking loss We argue that a ranking approach (*which reading is more plausible?*) is more suitable for addressing our problem compared with a classification approach (*plausible vs. implausible*). The reason is that ranking allows for a more relaxed and graded notion of textual plausibility: we want the model to *prefer* one variant and not to *choose* one variant. This is accomplished by reducing the margin ranking loss on the training data $\{(r_i^+, r_i^-)\}_{i=1}^n$:

$$\mathcal{L}_\theta = \frac{1}{n} \sum_{i=1}^n \left[1 - score_\theta(r_i^+) + score_\theta(r_i^-) \right], \quad (1)$$

where $score(\cdot)$ is a plausibility prediction model parameterized by θ . The plausibility-prediction model which we use is described in detail in the following paragraphs. Since \mathcal{L}_θ is differentiable with respect to the model’s parameters θ , we can learn them with gradient descent.

Model overview We desire the $score(\cdot)$ function to return a number $p \in \mathbb{R}$ reflecting the plausibility

of a text sequence made up of words w_1, \dots, w_n . In our case, this function is instantiated with (i) a Siamese reading encoder (*Reading Encoder*, Figure 2) and a Siamese plausibility prediction layer for producing a plausibility score for any given text (*Plausibility Prediction*, Figure 2). Now, we will describe these two components more closely.

Reading encoder First, we use a contextual language model⁵ to infer a sequence of word embeddings: e_1, \dots, e_n , which correspond to words w_1, \dots, w_n . Here, we hope that already the contextual language model provides statistical information indicating whether a specific word sequence may be considered as rather plausible or rather implausible (‘inductive bias’). The sequence of word embeddings e_1, \dots, e_n is further multiplied by a sequence of positive indicator coefficient embeddings: $e_1 \cdot c_1, \dots, e_n \cdot c_n$.⁶ This allows the model to learn to better distinguish between the source, target and the connector text (we learn three corresponding indicator embeddings). The resulting sequence is further processed by (ii) a Bi-LSTM (Hochreiter and Schmidhuber, 1997) to construct hidden states $H = h_1, \dots, h_n$ (we concatenate hidden states of forward and backward read) and (iii) a four-headed scaled dot-product self-attention mechanism (Vaswani et al., 2017), where in our case we use $H = Q = K = V$:

$$\begin{aligned} Heads(Q, K, V) &= [head_1; \dots, head_4]W^O \\ head_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \\ \text{Attention}(Q, K, V) &= \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \end{aligned}$$

where $W_{(\cdot)}^{(\cdot)}$ are parameters of the model. Finally, we compute a weighted average of the final sequence of hidden states to construct a vectorized reading representation v (Felbo et al., 2017):

$$\begin{aligned} e_t &= Heads(\cdot)_t W^A & a_t &= \frac{\exp(e_t)}{\sum_{i=1}^T \exp(e_i)} \\ v &= \sum_{i=1}^T a_i Heads(\cdot)_i, \end{aligned}$$

where $Heads(\cdot)_t$ is the vector corresponding to time step t computed by the previous scaled dot-

⁵We use BERT (Devlin et al., 2019) to infer the contextual embeddings. In our ablation experiments, we also present results based on ELMo embeddings (Peters et al., 2018)

⁶Similar to Opitz and Frank (2019a).

abbreviation	‘support’	‘attack’
A/A	Additionally,	Admittedly,
A/D	I agree,	I disagree,
M/H	Moreover,	However,
Y/N	Yes,	No,

Table 1: Argumentative discourse connector sentence adverbials and the argumentative relation class which they are likely to signal.

product attention step and v is a final vectorized representation of the input reading.

Plausibility prediction At plausibility prediction time, the vector representation v , which we obtained by the previous step, is mapped to a single score by means of a linear combination with a weight vector. Lastly, a selu-function (Klambauer et al., 2017) produces the desired plausibility-score:

$$p = \text{selu}(v^T w). \quad (2)$$

This score, computed once for each of the two competing reconstructions, allows a comparison with respect to their (predicted) plausibility. For our ARC experiments, where we desire a final classification, we predict the argumentative relation class by inspecting the discourse connector of the reconstruction which obtains a higher plausibility score. E.g., if $\text{score}(\text{EAU}_1, \text{additionally}, \text{EAU}_2) \geq \text{score}(\text{EAU}_1, \text{admittedly}, \text{EAU}_2)$ we predict the argumentative ‘support’ relation – otherwise we predict the ‘attack’ relation.

4 Experiments

We begin this section by describing the experimental setup used to evaluate our neural plausibility ranker. Next, we present our main results and finally perform several analyses and study the effects of ablating model components.

4.1 Setup

Discourse links To construct plausible and implausible texts, we experiment with eight different discourse connectors which have the potential to ‘signal’ argumentative relation types. They make up, in total, four minimal pairs (Table 1).

Data We use the student essay corpus v02 (Stab and Gurevych, 2017) in two versions: ESSAY and ESSAY-CONTENT. What is common to both is that

they contain data from the same 402 argumentative essays written by students about a variety of topics. The essays have been annotated with, i.a. spans of argumentative units and their relations with each other (support vs. attack). Since only the argumentative clauses have been annotated, we can clean EAUs from their discourse context, which yields ESSAY-CONTENT. For example, consider EAU_1 . *To add on this, EAU₂*. While in ESSAY, a system is allowed to see EAU-surrounding tokens (*to add on this*), in ESSAY-CONTENT, systems are allowed to see only the spans of the EAUs to predict their relation (i.e., $\text{EAU}_1, \text{EAU}_2$). In the easy case, *to add on this* may be enough to predict a support relation with high confidence and accuracy without even seeing the content of the EAUs – in the hard case, however, a system must learn to assess the actual content of the premises. In ESSAY-CONTENT, the performance of the feature-based SVM described by Stab and Gurevych (2017) drops by more than 23% macro F1 compared to the standard setup (ESSAY) where shallow discourse context is accessible (Opitz and Frank, 2019b).

Baselines We display the results of a competitive feature based SVM. It requires, i.a., syntactic parsing, constituency-tree sentiment annotation (Socher et al., 2013) and discourse parsing (Lin et al., 2014) as pre-processing steps (Stab and Gurevych, 2017; Opitz and Frank, 2019b). In contrast, our method does not depend on any pre-processing.

Model instantiation For each possible minimal pair, we instantiate a different model based on the pre-trained BERT model (the BERT model remains fixed during optimization). More specifically, we infer the word embeddings and average over the last four layers to produce a sequence of vectors with 1024 dimensions. Forward and backward LSTM have 256 neurons each. For development purposes we split off 1149 examples from the training data. The rank loss (Eq. 1) is minimized by performing stochastic gradient descent with Adam (Kingma and Ba, 2014)⁷. After each epoch, the model is evaluated on the development data. Finally, we select the parameters from the epoch with maximum F1 score on the development data.

In our tables, each model is denoted ArgRanker_{dcs} where dcs indicates which pair of discourse connectors was used for reconstruc-

⁷The learning rate is set to 0.001, the mini-batch size to 64 and the maximum number of epochs to 25.

System	ESSAY	ESSAY-CONTENT
majority baseline	47.8	47.8
SVM with features	68.0	57.3
ArgRanker _{A/A}	$67.2^{\pm 1.0}$	$58.6^{\pm 1.4}$
ArgRanker _{A/D}	$69.2^{\pm 2.4}$	59.2 $^{\pm 0.7}$
ArgRanker _{M/H}	$68.8^{\pm 1.7}$	63.8 $^{\pm 2.1}$
ArgRanker _{Y/N}	$67.3^{\pm 0.8}$	$58.3^{\pm 1.8}$
ArgRanker _{vote}	70.9 $^{\pm 0.7}$	60.7 $^{\pm 1.7}$

Table 2: Macro F1 results. underlined: best result; **bold**: improves against SVM withstanding standard deviation.

tion. $\text{ArgRanker}_{\text{vote}}$ denotes a model where we aggregate the predictions over the four different minimal-pair single models ('ensemble model'). All results are averaged over five runs.

4.2 Results

Macro F1 results Table 2 lists the macro F1 results⁸ of our experiments.

On ESSAY, our method is competitive with the SVM that relies on extensive pre-processing. On ESSAY-CONTENT, where models are forced to learn to assess the content of EAUs, our method outperforms the feature-based SVM across all configurations. The best performance on this data is provided by ArgRanker_{M/H}, which is trained on *Moreover-However* reconstructions (+6.5 pp. macro F1, relative improvement: 11%). Our ensemble model ArgRanker_{vote}, which aggregates the predictions of the individual ArgRankers in a simple vote, achieves an improvement of +3.4 pp. macro F1 (relative improvement: 6%).

More detailed results Table 3 indicates that our method offers other advantages besides raw macro F1 gains. The very rare *attack*-class is detected with a much greater precision compared with the SVM. The difference can range from an improvement of 5.6 pp. (ArgRanker_{Y/N}, relative improvement: 28%) up to a maximum improvement of 31.4 pp. (ArgRanker_{vote}, relative improvement: 157%). With such a large increase in precision, one might expect a drop in recall – however, this is only the case to a very small extent. The greatest drop in recall is incurred by ArgRanker_{vote} (-5.2 pp.) and thus can be said to lie in the shadow of its precision gains (+31.4 pp.). Moreover, when

⁸Macro F1 in our case is defined as the unweighted mean over the F1 scores for our two classes.

System	Attack		Support	
	Precision	Recall	Precision	Recall
majority	0.00	0.00	8.0	<u>100.0</u>
SVM with features	20.0	22.9	93.0	91.8
ArgRanker _{A/A}	31.0 $^{\pm 5.9}$	$18.2^{\pm 2.1}$	$92.9^{\pm 0.1}$	96.2 $^{\pm 1.1}$
ArgRanker _{A/D}	28.0 $^{\pm 4.4}$	$22.8^{\pm 3.2}$	$93.1^{\pm 0.2}$	94.5 $^{\pm 1.6}$
ArgRanker _{M/H}	39.9 $^{\pm 9.4}$	30.0 $^{\pm 6.8}$	93.8 $^{\pm 0.6}$	95.5 $^{\pm 2.4}$
ArgRanker _{Y/N}	$25.6^{\pm 5.8}$	$23.7^{\pm 5.1}$	93.1 $^{\pm 0.3}$	$93.0^{\pm 3.3}$
ArgRanker _{vote}	51.4 $^{\pm 7.3}$	$17.7^{\pm 3.4}$	$93.0^{\pm 0.2}$	98.4 $^{\pm 0.7}$

Table 3: Precision and recall scores for each class on ESSAY-CONTENT. underlined: best result; **bold**: improves against SVM withstanding standard deviation.

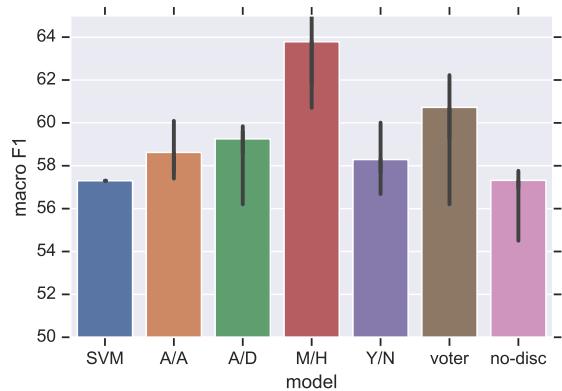


Figure 3: Scores for different models using BERT embeddings and SVM (left column) on ESSAY-CONTENT. Reconstruction with *Moreover/however* offers the largest improvement. Non-linguistically motivated connectors result in reduced performance ('+'/'-': *no-disc*, right column).

we use the discourse connector minimal pairs A/D and M/H, our model outperforms the SVM in the *attack*-class both in precision and recall. Most notably, when we instantiate our reconstructions with *Moreover/However*, we see a large gain in precision (+19.9 pp., relative improvement: 99.5%) but also an observable gain in recall (+7.1 pp., relative improvement: 31.0%).

With regard to the majority class (*support*), we make two observations: (i) precision-wise, all of our models outperform or are on par with the SVM; (ii) recall-wise, all of our models outperform the SVM. The greatest gain in recall for *support* is achieved by ArgRanker_{vote} (+6.6 pp.).

4.3 Ablation experiments and analysis

Linguistically motivated discourse reconstruction What is the outcome of instantiating the dis-

System	model configuration			
	basic	ELMo	-coeff.	-att.
majority	47.8	-	-	-
SVM (Stab and Gurevych, 2017; Opitz and Frank, 2019b)	57.3	-	-	-
ArgRanker _{A/A}	58.6 ^{±1.4}	55.7 ^{±1.6}	57.4 ^{±2.1}	58.3 ^{±1.2}
ArgRanker _{A/D}	59.2 ^{±0.7}	60.2 ^{±2.2}	59.6 ^{±2.1}	56.2 ^{±1.9}
ArgRanker _{M/H}	63.8 ^{±2.1}	59.4 ^{±2.5}	61.1 ^{±1.0}	60.7 ^{±1.9}
ArgRanker _{Y/N}	58.3 ^{±1.8}	57.6 ^{±1.7}	57.7 ^{±1.2}	58.2 ^{±3.0}
ArgRanker _{vote}	60.7 ^{±1.7}	59.5 ^{±1.9}	60.2 ^{±2.2}	56.2
ArgRanker _{-discourse}	57.3 ^{±0.4}	63.6 ^{±1.3}	57.3 ^{±2.8}	54.5

Table 4: Ablation experiments: Macro F1 results on ESSAY-CONTENT. *ArgRanker_{-discourse}*: a system where we replace the natural discourse connectors with ‘linguistically meaningless’ placeholders (i.e., support: ‘+’, attack: ‘-’ instead of, e.g., support: ‘Moreover’, attack: ‘However’). *ELMo*: we use ELMo instead of BERT; *-coeff.*: we abstain from learning source-target specific coefficients; *-att.*: we ablate the self-attention and use the last states of the Bi-LSTM (concatenation of each read) for prediction.

course reconstructions with ‘meaningless’ connectors? I.e., instead of instantiating the attack/support context with linguistically motivated connectors, such as, e.g., *I agree/I disagree*, we instantiate the contexts with the meaningless tokens ‘+’ and ‘-’. On one hand, this means that the new discourse configuration is still discriminative (either supporting or attacking). On the other hand, however, the discriminating reconstruction is not any more linguistically motivated. Thus, we hypothesize that the linguistically motivated reconstructions better ‘trigger’ the contextual BERT model into giving a useful inductive bias about whether a certain reading is plausible or not.

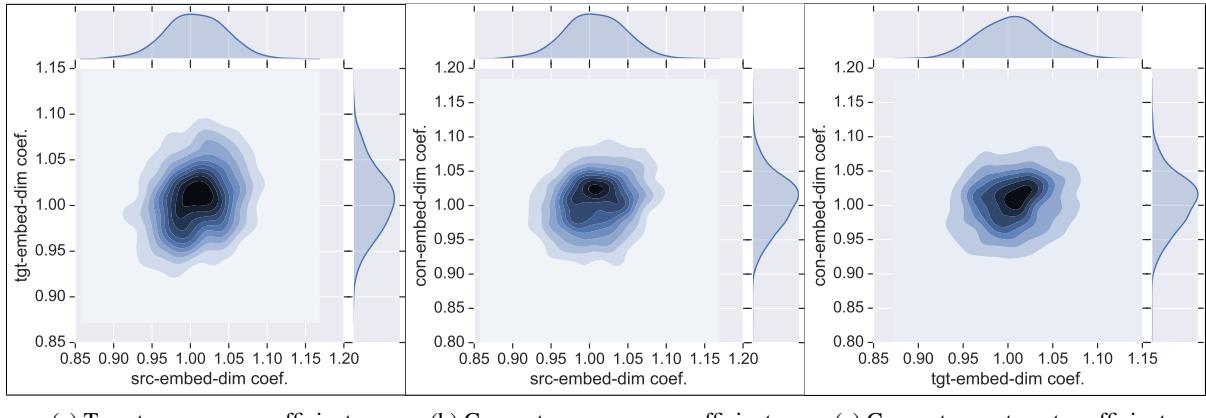
From Table 4 and Figure 3, we see that, indeed, our model functions better when provided with linguistically motivated reconstructions instead of the non-linguistically motivated reconstruction (Figure 3: columns *A/A*, *A/D*, *M/H*, *Y/N* vs. bottom row in Table 4 and **right column** in Figure 3). This holds true across all model configurations and all linguistically motivated discourse connector pairs.⁹

More specifically, we find that the *Moreover/However* reconstruction appears to offer the most useful inductive bias (**middle column**, Figure 3). Our *ArgRanker* based on this reconstruction outperforms all other configurations by more than 4 pp. macro F1 (compared with *Agree/Disagree*) and more than 6 pp. macro F1 compared with the

non-linguistically motivated reconstruction. One reason could be located in the fact that BERT was trained, i.a., on the Wikipedia corpus: we compute a simple word frequency statistic over this corpus and see that the terms *Moreover* and *However* appear more frequently in this corpus (e.g., *however*: appr. 29,900,000 occurrences) than, e.g., *Admittedly* (appr. 17,000 occurrences). Also, by manually inspecting a small amount of occurrences in Wikipedia, we find that *moreover* and *however* tend to occur in more ‘argumentative’ contexts, or, at least, connect two discourse units in a contrasting (*however*) or supporting (*moreover*) way. On the other hand, e.g., *I agree* tends to occur in less argumentative contexts, such as in *I agree to the terms of service*. We believe that contextual language models trained on interactive discourse texts (e.g., online discussion platforms) instead of encyclopedic texts would greatly help to provide our model with better embeddings in the situations where we want to compose plausible and implausible texts by means of more ‘interactive’ connectors (*I agree/I disagree*; *Yes/No*; etc.).

BERT vs. ELMo In our first experiment, we replace the BERT embeddings with ELMo embeddings – we want to ‘probe’ which of the two embedding generators is better suited to rank argumentative texts according to their plausibility. First, we see that ELMo embeddings provide better performance than the feature based baseline, with one exception: ArgRanker_{A/A}, where we reconstruct contexts by inserting *Additionally* and *Admittedly*

⁹An exception constitutes the model based on ELMo embeddings, which appears to work better when provided with the non-linguistically motivated connector pair.



(a) Target vs. source coefficients (b) Connector vs. source coefficients (c) Connector vs. target coefficients

Figure 4: Investigation of the three contextual coefficient-embeddings which our model has learned. The coefficients are initialized with ones and assume, during training, a Gaussian-like distribution. By all appearances, the model uses some coefficients to ‘deflate’ the impact of an embedding dimension in, e.g., the text corresponding to the source EAU, while ‘inflating’ the impact of an embedding dimension in, e.g., the text corresponding to the target EAU (Figure 4a, regions on the upper left).

(Table 4, **ELMo**). Second, the ELMo embeddings in most cases fall short in comparison to BERT embeddings – again, however, with one exception: ArgRanker_{discourse}, which does not use the linguistically motivated reconstruction.

Indicator embedding coefficients Now, we want to investigate if learning coefficients to better distinguish between source and target has helped our model. Recall, that the three coefficient indicator embeddings correspond to *source/target EAU span* and the *discourse connector span* and allow the model to highlight certain word embedding indices differently with respect to these three spans. For most connector pairs, learning the coefficients helps and their ablation leads to a performance drop (Table 4, **-coeff**; e.g. ArgRanker_{M/H}: -2.7 pp. macro F1).

Finally, we plot the learnt coefficients of the three different indicator embeddings against each other to analyze their appearance after training. Figure 4 displays all values from all discourse connector parameterizations \cdot/\cdot of ArgRanker_{./..}. More specifically, we are interested in the following question: *Have we learned that certain contextual word embedding indices are important to inflate (deflate) with respect to the source or the target?* From inspecting Figure 4, we see that this appears to be the case. For example, there is a set of embedding indices where coefficients are used to magnify the corresponding values in the target EAU and deflate them in the source EAU (Figure 4a, top left region) – while for another set of embedding indices

the opposite is true (Figure 4a, bottom right). Furthermore, the learnt coefficients have assumed a normal-like distribution after training (distribution plots on the sides of Figures 4a, 4b, 4c).

Self-attention Finally, we want to investigate the effect of ablating the self-attention mechanisms from our model. More precisely, we predict the plausibility scores based on a concatenation of the last state of forward and backward read of the Bi-LSTM. Throughout all different discourse reconstruction strategies, we see drops in performance (Table 4, **-att**). However, while we see observable drops in some cases (ArgRanker_{vote}: -4.5 pp. macro F1), they are comparatively small in other cases (ArgRanker_{Y/N}: -0.1 pp.).

5 Conclusion

We have treated argumentative relation classification in a new light, as a task where we learn to rank candidate texts according to their plausibility. To this aim, we have proposed a simple reconstruction trick which allows us to embed source and target argumentative units into plausible and implausible argumentative discourse contexts. In order to learn to rank such texts according to their plausibility, we have adapted a neural Siamese ranking model. Our experiments on an established data set have shown that the approach is competitive with previous work albeit it does not require pre-processing. In the ‘content-based’ setting – which is more difficult because models cannot base their decisions on shallow clues in the discourse context – the method

outperforms previous work by a considerable margin. In particular with respect to the scarce class *attack* we observed substantial improvements in precision.

Acknowledgments

We are grateful to Anette Frank for valuable discussions and feedback on an earlier draft of this paper. This work has been supported by the Leibniz ScienceCampus “Empirical Linguistics and Computational Language Modeling”, supported by the Leibniz Association grant no. SAS-2015-IDS-LWC and by the Ministry of Science, Research, and Art of Baden-Württemberg.

References

- Philippe Besnard and Anthony Hunter. 2014. Constructing argument graphs with deductive arguments: a tutorial. *Argument & Computation*, 5(1):5–30.
- Katarzyna Budzynska, Mathilde Janier, Juyeon Kang, Chris Reed, Patrick Saint-Dizier, Manfred Stede, and Olena Yaskorska. 2014. Towards argument mining from dialogue. In *COMMA*, pages 185–196.
- Oana Cocarascu and Francesca Toni. 2017. Identifying attack and support argumentative relations using deep learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1374–1379.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.
- Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625, Copenhagen, Denmark, September. Association for Computational Linguistics.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Yufang Hou and Charles Jochim. 2017. Argument relation classification using a joint inference model. In *Proceedings of the 4th Workshop on Argument Mining*, pages 60–66.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. *CoRR*, abs/1706.02515.
- Jonathan Kobbe, Juri Opitz, Maria Becker, Ioana Hulpus, Heiner Stuckenschmidt, and Anette Frank. 2019. Exploiting Background Knowledge for Argumentative Relation Classification. In Maria Eskevich, Gerard de Melo, Christian Fäth, John P. McCrae, Paul Buitelaar, Christian Chiarcos, Bettina Klimek, and Milan Dojchinovski, editors, *2nd Conference on Language, Data and Knowledge (LDK 2019)*, volume 70 of *OpenAccess Series in Informatics (OASIcs)*, pages 8:1–8:14, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2.
- Anne Lauscher, Goran Glavaš, and Kai Eckert. 2018a. Arguminsci: A tool for analyzing argumentation and rhetorical aspects in scientific writing. In *Proceedings of the 5th Workshop on Argument Mining*, pages 22–28.
- Anne Lauscher, Goran Glavaš, and Simone Paolo Ponzetto. 2018b. An argument-annotated corpus of scientific publications. In *Proceedings of the 5th Workshop on Argument Mining*, pages 40–46, Brussels, Belgium, November. Association for Computational Linguistics.
- Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning, KR’12*, pages 552–561. AAAI Press.
- Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184.
- Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology (TOIT)*, 16(2):10.
- Ana Marasović, Leo Born, Juri Opitz, and Anette Frank. 2017. A mention-ranking model for abstract anaphora resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 221–232, Copenhagen, Denmark, September. Association for Computational Linguistics.
- Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Thirtieth AAAI Conference on Artificial Intelligence*.

- Juri Opitz and Anette Frank. 2018. Addressing the Winograd schema challenge as a sequence ranking task. In *Proceedings of the First International Workshop on Language Cognition and Computational Models*, pages 41–52, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.
- Juri Opitz and Anette Frank. 2019a. An argumentative marker model for syntax-agnostic proto-role labeling. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 224–234, Minneapolis, Minnesota, June. Association for Computational Linguistics.
- Juri Opitz and Anette Frank. 2019b. Dissecting content and context in argumentative relation analysis. In *Proceedings of the 6th Workshop on Argument Mining*, pages 25–34, Florence, Italy, August. Association for Computational Linguistics.
- Andreas Peldszus and Manfred Stede. 2013. From argument diagrams to argumentation mining in texts: A survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31.
- Andreas Peldszus and Manfred Stede. 2015. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948.
- Andreas Peldszus and Manfred Stede. 2016. An annotated corpus of argumentative microtexts. In D. Mohammed and M. Lewinski, editors, *Argumentation and Reasoned Action - Proc. of the 1st European Conference on Argumentation, Lisbon, 2015*. College Publications, London.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June. Association for Computational Linguistics.
- John L. Pollock. 1995. *Cognitive Carpentry: A Blueprint for How to Build a Person*. MIT Press, Cambridge, MA, USA.
- Maria Skeppstedt, Andreas Peldszus, and Manfred Stede. 2018. More or less controlled elicitation of argumentative text: Enlarging a microtext corpus via crowdsourcing. In *Proceedings of the 5th Workshop on Argument Mining*, pages 155–163, Brussels, Belgium, November. Association for Computational Linguistics.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56.
- Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.
- Reid Swanson, Brian Ecker, and Marilyn Walker. 2015. Argument mining: Extracting arguments from online dialogue. In *Proceedings of the 16th annual meeting of the special interest group on discourse and dialogue*, pages 217–226.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.
- Henning Wachsmuth, Manfred Stede, Roxanne El Baff, Khalid Al-Khatib, Maria Skeppstedt, and Benno Stein. 2018. Argumentation synthesis following rhetorical strategies. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3753–3765, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.
- Douglas Walton. 2009. Objections, rebuttals and refutations. In *Argument Cultures: Proceedings of the 8th OSSA Conference*, pages 1–10, Windsor, Ontario.

Predicting Semantic Labels of Text Regions in Heterogeneous Document Images

Somtochukwu Enendu

University of Twente

Enschede, The Netherlands

senendu5@yahoo.com

Johannes Scholtes

ZyLAB

Amsterdam, The Netherlands

Johannes.Scholtes@zylab.com

Jeroen Smeets

ZyLAB

Amsterdam, The Netherlands

Jeroen.Smeets@zylab.com

Djoerd Hiemstra

Radboud University Nijmegen

Nijmegen, The Netherlands

Djoerd.Hiemstra@ru.nl

Mariet Theune

University of Twente

Enschede, The Netherlands

m.theune@utwente.nl

Abstract

This paper describes the use of sequence labeling methods in predicting the semantic labels of extracted text regions of heterogeneous electronic documents, by utilizing features related to each semantic label. In this study, we construct a novel dataset consisting of real world documents from multiple domains. We test the performance of the methods on the dataset and offer a novel investigation into the influence of textual features on performance across multiple domains. The results of the experiments show that the neural network method slightly outperforms the Conditional Random Field method with limited training data available. Regarding generalizability, our experiments show that the inclusion of textual features aids performance improvements.

1 Introduction

On a daily basis, legal departments of corporations produce many electronic documents for documentation of cases, investigative reporting, internal communication etc. Whenever these corporations are involved in litigation or investigations as part of regulatory requests, the need arises to collect and review these documents and share their contents with third parties. As document data sets increase, the corporations turn to e-discovery technology to facilitate the process of collecting, reviewing and sharing. E-discovery technology helps to automatically analyze the documents by using text mining and other text-related analytics to discover relevant information. However, these text mining techniques for automatic document analysis only work

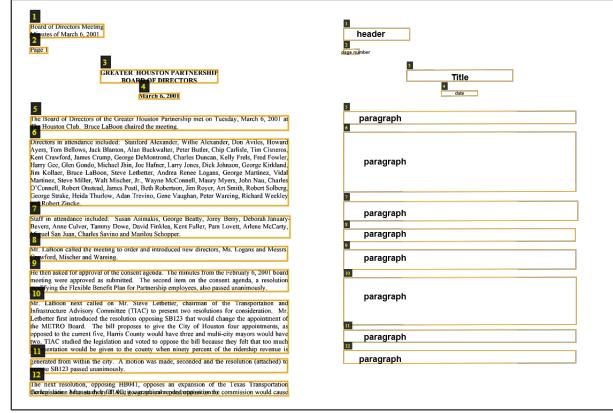


Figure 1: Example of a segmented document and its corresponding labels

optimally when the roles of different text sections in a document are known. For example, by recognizing tables, headers and footers, we can apply different extraction and analysis techniques than on normal paragraphs, and expect better results.

For safety reasons however, electronic documents in the legal domain are mostly transformed into images (e.g. jpg, tiff) so the corporation or firm can have control of what they share with other parties. Electronic documents usually contain hidden information (information that can't be seen when the document is viewed) and these pieces of information could contain hidden details they don't want to disclose to the receiving party. On the other hand, transforming the documents to images creates another problem as it makes it more difficult to automatically identify the specific role of the document areas. Hence, to provide automatic tools to determine the function of textual regions derived from document images, we need to do document image understanding.

The primary goal in document image understand-

ing is to (1) identify regions of interest in a document image (page segmentation) and (2) recognize the role of each region (semantic structure labeling). Many related studies treat these two tasks as separate sequential tasks. However, they are also often handled as one unified task. In this work, we specifically address the second step in the understanding of document images: the task of semantic structure labeling. The goal of this task is to label a sequence of physically segmented regions in a document image with semantic labels such as header, paragraph, footer, caption, etc. (see Figure 1). We treat the task as a sequence labeling problem, which involves assigning a categorical label to each member of a sequence of observations i.e. a sequence of document segments in our scenario. Though the work of document image understanding covers various types of document images, our work focuses on electronic and digital-born documents composed primarily of single-column layouts. Typical examples of such electronic documents which can be converted to images are PDF, Word, Powerpoint, E-mails, etc.

Even though extracting the semantic information from a document is a task that is easily done by a human, it is still an open and challenging problem for computers due to the inherent complexity of documents (Rangoni et al., 2012), especially when the set of documents in focus are diverse in layout and format. Similar works on semantic labeling such as (Tao et al., 2013) and (Shetty et al., 2007) are usually very specific to a document format or a set of related document types and problematic when we try to generalize to other document types. There is still a need for robust methods, capable of dealing with a broad spectrum of layouts found in digital-born documents (Clausner et al., 2011).

Our work addresses this gap in research by comparing sequential labeling methods for the semantic labeling task, and considering heterogeneous document images. Homogenous formats and lack of fine-grained semantic labels relevant for real world documents, are some limitations of previous document image datasets. To address these issues, we annotated a new dataset containing documents from an infamous legal case - the Enron Corporation scandal investigation. We also compare the performance of the following sequence labeling methods on the annotated dataset: (i) A feature-based Conditional Random Field (CRF) (ii) A recurrent neural network with a Bidirectional

Long Short-Term Memory (LSTM) architecture.

Our methods perform fine-grained recognition on text regions and include identification of tables. Furthermore, we check the influence of textual related features on the generalizability of our methods to a different domain. Luong et al. (2010) and Yang et al. (2017) prove that the performance of methods improves when text information in a region is considered for semantic labeling. We extend this by checking its influence across a different document domain.

Our main contributions are summarized as follows:

- We compare two sequential labeling methods to address document semantic structure labeling. Unlike previous works, we consider heterogeneous document formats and identify both fine-grained semantic-based classes and tables.
- We offer a novel investigation into the influence of text-related features on the performance of our methods across a different document domain.
- We provide an evaluation dataset for the task of semantic labeling on digital-born documents.¹

In section 3, we present our evaluation dataset. We then provide a detailed description of our system architecture in section 4. Section 5 is a breakdown of the sequence labeling methods performed for the task. We show the results of our experiments in section 6 and conclude on our work in section 7.

2 Related Work

Previous works on document image understanding (Chen and Blostein, 2007; Marinai, 2008; Kamola et al., 2015) divide the task into two parts: a physical decomposition or segmentation of document images into regions (page segmentation) and a logical/semantic understanding of these regions (semantic structure labeling). Though the focus of our work is on semantic labeling, we also present a high-level discussion on existing page segmentation techniques.

¹The dataset will be made available upon request.

2.1 Page Segmentation

Page segmentation techniques involve identifying segments enclosing homogeneous content regions, such as text, table, figure or graphic in a document page or image. These techniques fall into three categories: *bottom-up*, *top-down* and *hybrid* approaches. Bottom-up approaches (Kise et al., 1998; Adnan and Ricky, 2011) begin by grouping pixels of interest and merging them into larger blocks or connected components, which are then clustered into words, lines or blocks of text. However, such approaches are expensive from a computational point of view. Top-down approaches (Antonacopoulos, 1998; Gatos et al., 1999) recursively segment large regions in a document into smaller sub regions. Both approaches however, are limited by their inability to successfully segment complex and irregular document layouts. Hybrid methods, such as proposed in Pavlidis and Zhou (1992) combine both top-down and bottom-up techniques. With recent advances in deep neural networks, neural based models have become state-of-the-art for segmentation. Siegel et al. (2018) utilized a neural network to extract figures and captions from scientific documents. Yang et al. (2017) proposed a unified convolutional model to classify pixels in a document based on their visual appearance and underlying text content.

2.2 Semantic Structure Labeling

Our work focuses on the second aspect of document image understanding. Semantic labeling couples semantic meaning to a physical region or zone of a document after it has been segmented. Two types of approaches have been considered in the literature to handle this task: the *model-driven approach* and the *data-driven approach* (Mao et al., 2003). Early work in semantic structure labeling focused on the model driven approach. Models made up of rules, or trees, or grammars contained all the information that was used to transform a physical structure into a logical or semantic one. Rule based systems (Kim et al., 2000), though fast and human-understandable proved to be poorly flexible and unable to handle irregular cases and varying layouts.

Recent studies have considered the data-driven approach using supervised learning methods as an alternative to avoid the inflexibility and rigidity of manually built rule systems and mechanisms. These data-driven approaches make use of raw

physical data to analyze the document and no knowledge or predefined rules are given. Various document image datasets have been created for this purpose including images in the document space of electronic documents, scanned documents, magazines, newspapers etc. (Todoran et al., 2005; Antonacopoulos et al., 2009) but they are usually confined to a single domain or class. Chen et al. (2007) define a document space as the set of documents that a classifier is expected to handle. The labeled training and test samples are all drawn from this document space. Our dataset includes heterogeneous formats of electronic documents such as Microsoft Office files, PDF and email files which cover multiple domains like business letters, articles, memos, forms, reports, invoices etc. that significantly vary in layout and content.

Most existing supervised learning methods for semantic labeling use CRF and deep neural network approaches. Tao et al. (2013) built a CRF model as a graph structure to label fragments in a document. Shetty et al. (2007) used CRFs utilizing contextual information to automatically label extracted segments from a document. Yang et al. (2017) and Stahl et al. (2018) used visual cues and deep learning methods to analyze documents. In this study, we treat the semantic structure labeling task as a sequential labeling problem where a document image is modeled as a sequence of regions. The motivation for this is to model spatial dependencies and possible transitions between the different regions. Shetty et al. (2007) model spatial inter-dependencies between sequential segments in documents. Luong et al. (2010) also treat their semantic labeling task as an instance of the sequential labeling problem. CRFs and recurrent neural networks are popular sequential learning methods for this type of modeling. We offer a comparison of these state-of-the-art methods for semantic labeling across heterogeneous document formats in this study.

Luong et al. (2010) report in their work that adding textual information to a CRF model for semantic labeling improves its performance. We build on this work by also checking the influence of textual information on the performance of our methods across different document domains.

3 Datasets

This section describes the construction of our evaluation dataset for the task of semantic labeling

Dataset	SemLab	PRIMA
Document images	400	478
Document space	Office docs, PDF & Email	Magazine
Label categories	13	9

Table 1: Overview of the datasets used in this study.

which we call SemLab (SemLab coined from Semantic Labeling). The documents we used were gathered from the Enron Corpus. This corpus is a large database of approximately 600,000 emails generated by 158 employees of the Enron Corporation and acquired by the Federal Energy Regulatory Commission, a United States federal agency, during its investigation after the company’s collapse.

To compare the performance of the sequence labeling methods across different domains, we used the PRIMA dataset of Antonacopoulos et al. (2009). Table 1 contains an overview of both datasets.

3.1 Dataset Creation

We select documents for our dataset from the email folder of the then CEO of Enron corporation. Of all the employees in the corporation, he received the most emails. The documents comprise of sent and received email messages in the folder as well as document attachments. For attached documents, we consider four formats of documents: Word, PDF, Excel and Powerpoint documents, and ignore other file formats in the folder. This selection of different document formats meets the *variety* characteristic of an ideal dataset as described in Antonacopoulos et al. (2006) because several classes of document pages are represented. In total, we select 100 email messages and 406 unique documents from the CEO’s email folder. With each document containing different pages, the full set we collected from the email folder contained 2,447 document pages.

After selection of the electronic documents, we converted them to TIFF images since document images are the focus of our work. The SemLab evaluation dataset is a random selection of 400 documents from the 2,447 document images, containing a total of 2,869 regions and their ground truth representation in CSV format (see section 3.3).

3.2 Document Semantic Labels

We attempt to identify 13 labels in a document: *paragraph*, *page header*, *caption*, *section heading*, *footer*, *page number*, *table*, *list item*, *title*, *email header*, *email body text*, *email signature* and *email footer*. Our choice of labels is specific to regions in a document that contain text. Hence we didn’t consider regions in a document that are devoid of text e.g. figure, image, graphic etc.

3.3 Annotation Process

Apart from the document images part of our dataset, we created the geometric hierarchical structure of each image (in CSV format) as ground truth for the dataset. We achieved this as follows: For each region, the corresponding bounding box was given in terms of its x and y coordinates on the document image. Each region was also given a label from the set of 13 labels we defined. The bounding box coordinates were defined by page segmentation using the Tesseract OCR engine² while the labeling of the regions was done manually. Tesseract OCR performs an automatic full page segmentation of the document image thereby producing the bounded regions in the document. We allowed for manual correction of the regions by the annotators in case of a faulty or overlapping region. In total, 5 non-domain experts took part in annotating the sample of 400 document images independently. Each document image was annotated by 3 annotators (fixed number).

To make the manual annotation effort easier for the annotators, we split the 400 documents into 40 groups i.e. 10 documents per group, so that they had the liberty to annotate a minimum of 10 documents and a maximum of 400 documents. We set up the process by providing the annotators with a simple image editor tool to manually correct the segmentation (by specifying imprecise region boundaries using a variety of drawing modes such as using rectangles or arbitrary polygons) and label each region in a document image. We pre-loaded the labels into a drop-down editor to improve annotation efficiency. Hence, the annotator only needed to select the labels from a drop-down. To ensure that the annotators understood the annotation task, we provided a user guide containing complete instructions on how to use the image editor tool and carry out the labeling of the regions.

We measured the Inter-Annotator Reliability

²github.com/tesseract-ocr/tesseract accessed 2019-06-09

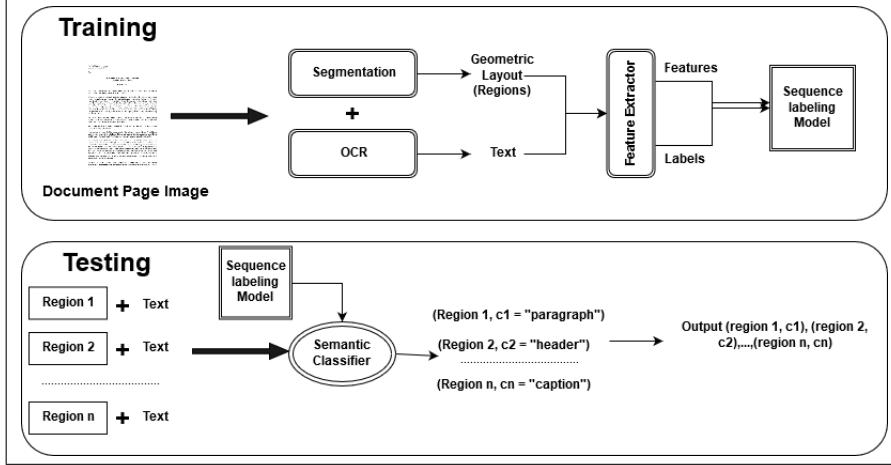


Figure 2: Implementation architecture, showing training and testing phases including the input and output for the sequence learning models

(IAR) of agreement using the Fleiss' Kappa measure (Fleiss, 1971). It has been shown to be more suitable to measure IAR when more than 2 annotators are involved, compared to other measures such as Cohen Kappa.³ The Fleiss' Kappa value measured for our annotation task was 0.52. This value indicates *moderate agreement* between the annotators, going by the table given in (Landis and Koch, 1977) for interpreting Fleiss' Kappa values. After annotation, the main author of this paper reviewed the annotations and resolved the disagreements between the three annotators for each document image. Disagreements were resolved by majority voting and in instances where each annotator had unique annotations, the author revisited the annotated samples and made the most logical choice of label to form the gold-standard set.

4 System Architecture

Figure 2 summarizes the architecture of our semantic labeling system. During the training process, we run all input document images through the Tesseract OCR software to obtain raw text data as well as geometric layout information. The feature extractor utilizes both the layout information and raw text, when available, to produce features which go through the sequence labeling trainer together with corresponding manually labeled data, to produce the learned models. The trainer learns to assign a semantic label to the segmented regions R of a document image D . Most of the document images contain single-column layouts, hence we order the

segmented regions as a sequence, from the top of the document page to the bottom. Each region $R_i \in R$ is bounded by a bounding box $B_i \in B$ that includes coherent text content and each bounding box is a set of pixels between its top left corner and bottom right corner coordinates. None of the bounding boxes overlap the other.

During testing, we want to assign a label $L_i \in W : i = \{1, \dots, n\}$ to each region R_i . Given a sequence of regions $x = (x_1, x_2, \dots, x_n)$ in a document image, the task is to determine a corresponding sequence of labels $y = (y_1, y_2, \dots, y_n)$ for x . This can be seen as an instance of a sequence labeling problem, which attempts to assign labels to a sequence of observations. We take into account the contextual information for each of the regions in the sequence i.e. the labels of preceding or following regions are taken into account for label classification.

5 Methods

In this section, we present the sequence labeling methods for semantic labeling of document images and the evaluation procedure.

5.1 Linear-Chain CRF (LC-CRF)

CRFs are probabilistic models used to segment and label sequential data. They are reported to be very effective for semantic structure detection (Peng and McCallum, 2004; Luong et al., 2010). An inherent merit of the CRF model to perform this task is its ability to combine two classifiers: a local classifier which assigns a label to the region based on local features and a contextual classifier to model contextual correlations between adjacent regions.

³Fleiss' Kappa works for any number of annotators giving categorical ratings, to a fixed number of items

Feature set	Description
Without OCR	
Block coordinates	The location of the region bounding box within the document image (x and y coordinates)
Height	Normalized height of block
Width	Normalized width of block
Area	Normalized area of block
Aspect ratio	Width/height of block
Vertical position	Vertical position of region in the image (top, middle, bottom)
With OCR	
Digit	Binary feature indicating if the text in the region consists of digits or contains digits
Capital letters	Binary feature indicating if the text in the region is all in capital case or contains capital letters
Nr of tokens	The number of tokens in a region block
Nr of lines	Binned number of lines in a region block (small, medium, large bins)
List item pattern	Binary feature indicating if text contains bullet items
Caption pattern	contains caption keywords (table, source, fig., figure)
Email keywords	Keywords found in different parts of an email
Has multi-white space (table feature)	Binary feature indicating if bounded region contains multiple white spaces between tokens.
% of white space (table feature)	The sum of white space lengths divided by the line length
Avg white space length (table feature)	The mean length of the white spaces within a line.

Table 2: Features used by the CRF methods.

Linear-chain CRFs are one well known type of CRFs which are similar to Hidden Markov Models but are reported to perform better (Peng and McCallum, 2004). They have one chain of connected labels. As CRF is a feature-based method, we implement two models with different feature sets in our work (see Table 2). We use the scikit-learn Python package, sklearn-crfsuite for implementation of our CRF models.

LC-CRF without OCR (LC-CRF₁): In this model, we exclude any features that can be extracted from the OCR output. That is, we consider only geometric/physical layout features to predict the label of a region in a document. The LC-CRF classifier will learn regions based on their position and location on the bounding box level of the document image. For example, it is common for *titles* to appear at the top of documents so the model may learn this observation from the extracted features.

LC-CRF with OCR (LC-CRF₂): By virtue of the generality and flexibility of CRF model, it is promising to achieve better performance by extending feature sets and exploring higher-level dependencies (Shetty et al., 2007). Luon et al. (2010) and Yang et al. (2017) report that by adding textual information to their models, there was an improvement in performance. We implement another LC-CRF model extending the feature set by including textual features from the OCR output. We also consider features for detecting tables. We re-use a subset of features for table detection in (Ghanmi and Abdel, 2014).

5.2 Recurrent Neural Networks (RNNs)

RNNs are a class of nets that are used for sequence learning. They can simultaneously take a sequence of inputs and produce a sequence of outputs. We transform the extracted feature sets of the CRF models into a 3D tensor and use this as input to the network. The shape of the 3D tensor is the number of input samples, the number of sequence regions per input sample and the number of features per sequence region. Therefore a shape of (300, 20, 30) indicates an input tensor of 300 document page samples, 20 regions per sample and 30 features for each region.

We use a Bidirectional-LSTM architecture for our network. Two neural models (RNN_1 and RNN_2) are trained and evaluated as such implemented for the CRF models, using feature sets with and without OCR features. Hyper-parameters are set in reference to the best performing configurations in Reimers and Gurevych (2017) with minor deviations. We use the adam algorithm for gradient descent optimization (Kingma and Ba, 2015). We don't include an embedding layer since we deal with numerical inputs, and set the number of recurrent units to 100 for all 3 hidden layers. Kernel and recurrent (l2) regularizers are added to our input layer. We introduce a batch normalization layer before the input layer and before each hidden layer to normalize the input values for our network. Normalizing or scaling the input values to a standard scale helps the network to learn the optimal parameters for each input node quickly and therefore, quickly find the minimum loss. Batch normalization also helps to improve the convergence properties of the network, has the effect of accelerating the training process of the network, and in some cases improves the performance of

	LC-CRF ₁	LC-CRF ₂	RNN ₁	RNN ₂
Overall Micro F_1	0.736	0.851	0.775	0.855
table	0.667	0.897	0.708	0.877
paragraph	0.617	0.811	0.622	0.774
page number	0.946	0.966	0.913	0.936
list item	0.336	0.594	0.559	0.697
heading	0.564	0.706	0.584	0.619
page header	0.868	0.914	0.846	0.865
title	0.571	0.703	0.677	0.747
footer	0.781	0.860	0.855	0.868
caption	0.667	0.742	0.742	0.771
email header	0.907	0.972	0.944	0.991
email body text	0.944	0.972	0.962	0.989
email signature	0.935	0.987	0.969	0.982
email footer	0.969	0.974	0.979	1.000

Table 3: Comparative performances among LC-CRF₁, LC-CRF₂, RNN₁ and RNN₂ models for semantic labeling. Category-specific performance given in F1. Results in bold mark the best system for each category.

the model. The inclusion of batch normalization layers in our network proves to be critical as it significantly improves performance. We add dropout regularization with a value of 0.1 to each hidden layer and use a batch size of 32 to control how often the weights of the network are updated. Furthermore, if the training loss does not decrease for 3 epochs, the learning rate is reduced by a 0.8 factor. Training is stopped if the minimum change in validation loss is less than 10^{-5} for 8 epochs or when 100 epochs are reached. We use the keras deep learning library running on top of tensorflow, for implementation of our RNN models.

5.3 Evaluation

The aim of our evaluation is to compare how sequence labeling methods perform for the task of semantic labeling of document regions and compare how their performances change with an extended feature set. We also evaluate the generalizability of our methods to a different document domain. Overall results are evaluated using the micro-averaged F_1 measure, the average of the results of 3 runs is reported per experiment. We split our dataset into train/test sets with a 70/30 ratio. We also perform 3-fold cross validation on the train set to tune the hyper-parameters of the model.

6 Results

6.1 Semantic Labeling of SemLab Dataset

Table 3 shows an overview of the results of our models comparison on the training dataset. The LC-

CRF model without OCR output (LC-CRF₁) performs fairly well, approaching an F_1 score of 0.74. It is clear however that including features from the OCR output has a significant impact: the LC-CRF₂ model with OCR increases micro-averaged F_1 to 0.85. We observe that including features from the OCR output also improves performance for the RNN method, with the RNN₂ model gaining a 0.8 increase compared to the RNN₁ micro-averaged F_1 score of 0.78. When contrasting the implemented methods, we see that the RNN method performs better than the LC-CRF method on both model variations. RNN₁ shows better F_1 scores than the LC-CRF₁ on the majority of the categories and the overall micro F_1 . The RNN₂ model also outperforms the LC-CRF₂ on most of the categories including the overall score. In addition, we make the following observations.

We observe that *list items*, *titles* and *headings* have the lowest scores for the best performing model. These categories usually have very similar features. For example, headings and list items are often started with numbering. Titles and headings also usually contain similar features such as having all capital letters. We also observe that list items have lower F_1 scores without OCR features. The classifier is able to only learn geometric and positional features of this category and misclassifies a lot of its samples as paragraph since both have similar locations on a document image and more so, paragraph is the majority category. The email related categories generally have high F_1 scores irrespective of the local feature sets included. This is because of the ability of sequence labeling methods to take into account the neighborhood of items; for example, an email body text is very likely to appear after an email header and thus the classifier learns this contextual knowledge.

6.2 Comparison across different document domain

In many real life scenarios, the datasets available to train models for the semantic labeling task are mainly homogeneous document images with similar or comparable layout and format. This raises the question about how generalizable a model that has been trained on a set or related set of document images is, to different domains. We trained the sequence labeling methods on our SemLab dataset which contains documents from multiple domains and tested each model on the records from the

Method	Testing Domain	
	SemLab	PRIMA
LC-CRF ₁	0.861	0.696
LC-CRF ₂	0.923	0.743
RNN ₁	0.888	0.701
RNN ₂	0.890	0.747

Table 4: Review of the transfer learning experiment. Each method is trained on the SemLab dataset and tested on in-domain and cross-domain documents. All scores are micro-averaged F_1 scores.

PRIMA dataset which contains documents from the magazine domain, not represented in our own dataset. For fair comparison, we evaluated only labels applicable to both datasets i.e. intersecting labels (header, paragraph, section heading, caption, page number, footer). For this reason we excluded some features from the ‘With OCR’ feature set that are directly related to the excluded labels.

Table 4 provides a summary of the performance of each method on the different domains. The results show that the methods have lower performances when evaluated on unseen data of a different domain than the training data. Both LC-CRF and RNN methods perform better when OCR information is included for the cross domain experiment. This proves that the inclusion of textual features also aids generalizability of methods across new domains for semantic labeling. Furthermore, we observe that both RNN methods are able to generalize better than the LC-CRF methods, though with slight improvements. This could be explained by the techniques specifically employed to reduce overfitting and improve generalizability power in the RNN such as the use of dropout, early stopping, l2 regularization, among others.

7 Conclusion and Future Work

In this work we have presented a comparison between state-of-the-art sequential learning models applied to the task of semantic labeling of document regions. We constructed a novel evaluation dataset to benchmark model performance on. The experimental results reveal that both methods are able to perform the task well using only a small amount of training data; with the RNN method slightly outperforming the LC-CRF method. Also, including OCR information in the feature set is promising to achieve better performance as it reduces confusion between ambiguous semantic classes. In addition, its inclusion

might positively affect generalization performance, as shown by our transfer learning experiments on the PRIMA domain.

Future work includes extending the document dataset in terms of size and variety to cover more document spaces, domains and classes. Models can exploit these characteristics to better generalize to new domains. By virtue of neural networks’ great power to learn latent features, we believe more (varying) data will also contribute to improving the performance levels of our neural method. An extension of the feature set used in this work could also be beneficial in improving performance scores for the implemented models.

References

- [Adnan and Ricky2011] Amin Adnan and Shiu Ricky. 2011. Page segmentation and classification utilizing bottom-up approach. *International Journal of Image and Graphics*, 01.
- [Antonacopoulos et al.2006] A. Antonacopoulos, D. Karatzas, and D. Bridson. 2006. Ground truth for layout analysis performance evaluation. In *Document Analysis Systems VII*, pages 302–311, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [Antonacopoulos et al.2009] A. Antonacopoulos, D. Bridson, C. Papadopoulos, and S. Pletschacher. 2009. A realistic dataset for performance evaluation of document layout analysis. In *2009 10th International Conference on Document Analysis and Recognition*, pages 296–300, July.
- [Antonacopoulos1998] A Antonacopoulos. 1998. Page segmentation using the description of the background. *Computer Vision and Image Understanding*, 70(3):350–369.
- [Chen and Blostein2007] Nawei Chen and Dorothea Blostein. 2007. A survey of document image classification: problem statement, classifier architecture and performance evaluation. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(1):1–16.
- [Clausner et al.2011] C. Clausner, S. Pletschacher, and A. Antonacopoulos. 2011. Scenario driven in-depth performance evaluation of document layout analysis methods. In *2011 International Conference on Document Analysis and Recognition*, pages 1404–1408, Sep.
- [Fleiss1971] Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.
- [Gatos et al.1999] B. Gatos, S. L. Mantzaris, K. V. Chandrinou, A. Tsigris, and S. J. Perantonis. 1999.

- [Integrated algorithms for newspaper page decomposition and article tracking. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99* (Cat. No.PR00318), pages 559–562, Sep.]
- [Ghanmi and Abdel2014] Nabil Ghanmi and Belaïd Abdel. 2014. Table detection in handwritten chemistry documents using conditional random fields. In *ICFHR*, pages p. 146–151, Crete, Greece.
- [Kamola et al.2015] Grzegorz Kamola, Michał Spytkowski, Mariusz Paradowski, and Urszula Markowska-Kaczmar. 2015. Image-based logical document structure recognition. *Pattern Anal. Appl.*, 18(3):651–665.
- [Kim et al.2000] Jongwoo Kim, Daniel X. Le, and George R. Thoma. 2000. Automated labeling in document images. In *Document Recognition and Retrieval*.
- [Kingma and Ba2015] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.
- [Kise et al.1998] Koichi Kise, Akinori Sato, and Motoi Iwata. 1998. Segmentation of page images using the area voronoi diagram. *Comput. Vis. Image Underst.*, 70(3):370–382.
- [Landis and Koch1977] J Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33 1:159–74.
- [Luong et al.2010] Minh-Thang Luong, Min-Yen Kan, and Thuy Dung Nguyen. 2010. Logical structure recovery in scholarly articles with rich document features. *Int. J. Digit. Library Syst.*, 1(4):1–23.
- [Mao et al.2003] Song Mao, Azriel Rosenfeld, and Tapas Kanungo. 2003. Document structure analysis algorithms: a literature survey. In *Document Recognition and Retrieval X, Santa Clara, California, USA, January 22-23, 2003, Proceedings*, pages 197–207.
- [Marinai2008] Simone Marinai, 2008. *Introduction to Document Analysis and Recognition*, pages 1–20. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Pavlidis and Zhou1992] Theo Pavlidis and Jiangying Zhou. 1992. Page segmentation and classification. *CVGIP: Graph. Models Image Process.*, 54(6):484–496.
- [Peng and McCallum2004] Fuchun Peng and Andrew McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2004, Boston, Massachusetts, USA, May 2-7, 2004*, pages 329–336.
- [Rangoni et al.2012] Yves Rangoni, Abdel Belaïd, and Szilárd Vajda. 2012. Labelling logical structures of document images using a dynamic perceptive neural network. *International Journal on Document Analysis and Recognition (IJDAR)*, pages 45–55.
- [Reimers and Gurevych2017] Nils Reimers and Iryna Gurevych. 2017. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *CoRR*, abs/1707.06799.
- [Shetty et al.2007] Shravya Shetty, Harish Srinivasan, Sargur Srihari, and Matthew Beal. 2007. Segmentation and labeling of documents using conditional random fields. *Proceedings of SPIE - The International Society for Optical Engineering*, 6500:6500–1.
- [Siegel et al.2018] Noah Siegel, Nicholas Lourie, Russell Power, and Waleed Ammar. 2018. Extracting scientific figures with distantly supervised neural networks. *CoRR*, abs/1804.02445.
- [Stahl et al.2018] Christopher Stahl, Steven Young, Drahomira Herrmannova, Robert Patton, and Jack Wells. 2018. Deepend: A deep learning approach to extracting text from pdfs. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).
- [Tao et al.2013] Xin Tao, Zhi Tang, and Canhui Xu. 2013. Document page structure learning for fixed-layout e-books using conditional random fields. *Proceedings of SPIE - The International Society for Optical Engineering*, 9021.
- [Todoran et al.2005] Leon Todoran, Marcel Worring, and Arnold W. M. Smeulders. 2005. The uva color document dataset. *International Journal of Document Analysis and Recognition (IJDAR)*, 7(4):228–240.
- [Yang et al.2017] X. Yang, E. Yumer, P. Asente, M. Kraley, D. Kifer, and C. L. Giles. 2017. Learning to extract semantic structure from documents using multi-modal fully convolutional neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4342–4351.

Evaluating Off-the-Shelf NLP Tools for German

Katrin Ortmann Adam Roussel Stefanie Dipper

Department of Linguistics
Fakultät für Philologie
Ruhr-Universität Bochum

ortmann|roussel|dipper@linguistics.rub.de

Abstract

It is not always easy to keep track of what tools are currently available for a particular annotation task, nor is it obvious how the provided models will perform on a given data set. In this contribution, we provide an overview of the tools available for the automatic annotation of German-language text. We evaluate fifteen free and open source NLP tools for the linguistic annotation of German, looking at the fundamental NLP tasks of sentence segmentation, tokenization, POS tagging, morphological analysis, lemmatization, and dependency parsing. To get an idea of how the systems' performance will generalize to various domains, we compiled our test corpus from various non-standard domains. All of the systems in our study are evaluated not only with respect to accuracy, but also the computational resources required.

1 Introduction

An extensive number of NLP tools are available nowadays for the automatic analysis of natural language data. The vast majority of these tools have been developed for English, though, and often take advantage of specific properties of the English language, such as the fact that English sentences show a rather fixed word order (so Markov models work well), or that English does not have a rich inflectional morphology (so lemmatizers are not a major concern). As a result, it is often not clear to what extent these tools are applicable to further languages, and often no pre-trained models are provided for languages other than English.

Similarly, these tools are mostly evaluated only on English language data, but the results for English may not be transferable to other languages. Depending on the language, different kinds of an-

notations can be necessary, which are not relevant for English.

Moreover, most tools are trained and tested on standard (newspaper) language, but different registers of a language can differ significantly, e.g. with respect to syntax or lexicon (Biber and Conrad, 2009). Performance of tools trained on standard language drops considerably when these tools are applied to data from other registers, such as social media data.

And finally, the efficiency of many freely available systems, i.e. the time and computing resources they require for the annotation task, can become a problem in the context of practical applications or if large amounts of text need to be analyzed.

The goals of this work are, first, to determine what freely-available systems exist for the linguistic analysis of German texts. Second, we want to assess the accuracy of their output in various non-standard domains, including formal (Wikipedia), semi-formal (sermons), and informal (movie subtitles) contexts. We will evaluate the systems in the fundamental NLP tasks of sentence segmentation, tokenization, part-of-speech (POS) tagging, morphological analysis, lemmatization, and dependency parsing, and we will provide an overview of the computational resources each system requires for these tasks.

The remainder of this paper is structured as follows: Section 2 gives an overview of related work. Section 3 introduces the data used in the study and the gold standard annotation. The NLP tools tested in this study are introduced in Section 4. In Section 5, we describe our experimental setup for the evaluation of the selected systems with respect to both accuracy and speed. We conclude with a discussion of the results in Section 6 and 7.

2 Related Work

The Association for Computational Linguistics (ACL) provides state-of-the-art results for a range

of NLP tasks, such as POS tagging, named entity recognition, parsing, paraphrase identification, or question answering.¹ However, the majority of results reported here are based on English data only, and even for common tasks like POS tagging, only results for English and French are available. This, of course, does not mean that POS tagging has not been evaluated for other languages as well. It shows, though, that results for other languages are scattered across numerous publications and not readily available. Moreover, even results of the same target language are not easily comparable to each other because evaluations often use different data sets.

While most evaluations are confined to newspaper texts, there are some exceptions. Gadde et al. (2011) evaluate and adapt a POS tagger trained on standard English data (the WSJ corpus) to SMS data. Gimpel et al. (2011) and Owoputi et al. (2012) develop POS taggers for Twitter data, with accuracies of around 90% (best version: 92.80%). Choi et al. (2015) evaluate statistical dependency parsers on the English part of OntoNotes 5, with data from different genres, which they specify as broadcasting conversation, broadcasting news, news magazine, newswire, pivot text, telephone conversation, and web text. In their study, the best overall accuracies are reached by the ClearNLP and the Mate parser with about 90%, while they found the spaCy parser (version 1.x) to be the fastest parser annotating 755 sentences or 13,963 tokens per second.

For German, Giesbrecht and Evert (2009) evaluate POS taggers on web data and show that accuracy values drop significantly for non-standard data, especially for posts from online forums, which were tagged with accuracies < 90%. Similarly, Neunerdt et al. (2013) evaluate and compare taggers trained on standard data with taggers retrained on comments in German from different online forums. Standard taggers achieve accuracies of > 87%, retrained taggers of > 93%. Beißwenger et al. (2016) show that these results hold true several years later, as they find similar results for the annotation of computer-mediated communication (CMC) and web data. While the best systems reach F_1 -Scores > 99% for tokenization, the best tagger only achieved a tagging accuracy of 90%.

¹https://aclweb.org/aclwiki/State_of_the_art

3 Data

As already mentioned, language varieties differ considerably on all linguistic levels, e.g. syntax or lexicon (Biber and Conrad, 2009), and pose different challenges to automatic annotation tools. Since most available models are trained on standard (newspaper) language, their accuracy might drop when they are applied to non-standard data like informal written communication. By using data from various domains, ranging from formal to informal contexts, we can evaluate whether the application of pre-trained models is limited to a restricted language domain or if they can be used for other language varieties as well.

For the evaluation, we used data from five different registers representing large resources of raw texts: encyclopedic text (*Wikipedia*)², literary text (*Nouvelette*)³, Christian sermons (*Sermon*)⁴, prepared but freely-performed talks (*TED*)⁵ and movie subtitles (*Movie*)⁶. Table 1 gives an impression of the selected text types and their particularities.

For each register, a random sample of approximately 1,500 tokens was selected, resulting in a total of 559 sentences with 7,642 tokens (2,649 types). Table 2 gives an overview of the data.

Gold Standard For all annotations we manually created a gold standard. The annotation of sentence boundaries⁷ and tokenization was carried out in text files, and all other annotations were done with WebAnno (de Castilho et al., 2016)⁸.

We use the Stuttgart–Tübingen tagset (or STTS, Schiller et al. (1999))⁹ for POS tagging, and the

²Sample `wpd15_sample.i5.xml` from the Wikipedia subcorpus of DeReKo (http://corpora.ids-mannheim.de/pub/wikipedia-deutsch/2015/wpd15_sample.i5.xml.bz2).

³Texts of the genre ‘nouvelette’ from GutenbergDE corpus, edition 14 (<http://gutenberg.spiegel.de/>), which were published after 1900.

⁴Automatically downloaded from the SermonOnline database (<http://www.sermon-online.de>)

⁵German translations of English talks, automatically downloaded from the official website <https://www.ted.com/talks?language=de>.

⁶German subtitles for movies tagged as “Action, Adventure, Drama” or “Comedy, Drama” from the OpenSubtitles corpus (<http://www.opensubtitles.org/>), downloaded at <http://opus.nlpl.eu/download.php?f=OpenSubtitles/v2018/raw/de.zip>

⁷For Wikipedia and Movie subtitles, sentence boundaries were already present in the data and only corrected as necessary.

⁸Version 3.4.6 (<https://webanno.github.io/webanno/>)

⁹STTS is the de facto standard POS tagset for German

Subcorpus	Example Sentence
Wikipedia	Das 6. Kanadische Kabinett (engl. <i>6th Canadian Ministry</i> , franz. <i>6e conseil des ministres du Canada</i>) regierte Kanada vom 21. Dezember 1894 bis zum 27. April 1896.
Novelle	Zwischen dem roten Rausch der Pelargonien und seinem Damaskus lag eine Fülle engbeschriebener Tagebuchblätter, jedes mit der zierlich-säuberlichen Unterschrift »Erich Friese, Kandidat« versehen.
Sermon	Wenn ihr aber zu Christus gehört, seid ihr auch Abrahams Nachkommen und bekommt das Erbe, das Gott Abraham versprochen hat. (Gal 3,29)
TED	Im Himalaya, der drittgrößten Eismasse, sehen Sie oben neue Seen, die vor ein paar Jahren Gletscher waren.
Movie	- Dad, das reicht!

Table 1: Example sentences from each register.

Subcorpus	Register	#Tok	#Sent	#Doc
Wikipedia	written, encyclopedic	1,514	96	12
Novelle	written, prose	1,588	69	12
Sermon	spoken, planned	1,520	90	16
TED	spoken, planned	1,506	101	17
Movie	spoken	1,514	203	21
Total		7,642	559	78

Table 2: Overview of the data from the five subcorpora used in the study.

TIGER annotation scheme (Crysmann et al., 2005) for morphological annotations, which can be considered an extension and improvement upon the original ‘large’ STTS. We also performed lemmatization on the texts, also according to the TIGER annotation scheme, albeit with some modifications: nominalizations are treated like normal nouns, and personal and reflexive pronouns are annotated with their nominative form (e.g. *ich*, *du*, etc.) except where this is not possible, as is the case for *sich*. For tokens that are not annotated with a lemma in the TIGER scheme, such as interjections, foreign words, punctuation, we use the surface form as its lemma.

For dependencies, we annotated only arguments, which should be reliably comparable across annotation schemes, and not modifiers, which are handled quite differently in different schemes, e.g. with regard to attachment site or label name.¹⁰ For this paper, we consider subjects, direct and indirect ob-

data. Morphological tagsets and lemmatization is less well standardized for German but most corpora use schemes that are based on or similar to the TIGER schemes we use.

¹⁰In order to be able to compare diverging annotation formats, we also annotated auxiliaries and prepositions within predicatives (cf. Section 5).

Annotation	#Tokens	#Types	#Ambig
POS	7,642	53	114
Lemma	7,642	2,077	27
Morphology	4,331	233	323

Table 3: Overview of the gold standard dataset.

jects, clausal subjects and objects, predicatives, and expletives to be arguments. We excluded prepositional objects because it is often difficult to determine when they are acting as arguments or adjuncts. All texts were annotated independently by one to five student annotators and checked manually afterwards by one of the authors.

Table 3 provides an overview of the gold data. The table shows the number of tokens with a particular type of annotation (which varies since words that cannot be inflected are not annotated for morphology), the number of annotated types, and the number of ambiguous word forms (e.g. the word form *gehört*, could either represent a present tense form of the lemma *gehören* ‘belong’ or a participle form of the lemma *hören* ‘hear’). Figure 1 shows the distribution of dependency relations in the gold data.

4 NLP Tools

For the evaluation in this paper, we selected systems that are freely available, i.e. open source, and for which pre-trained models for German are provided. Of course, the selection of tools we test here is not comprehensive, but we will make our data and evaluation scripts publicly available so further systems can be added and compared with

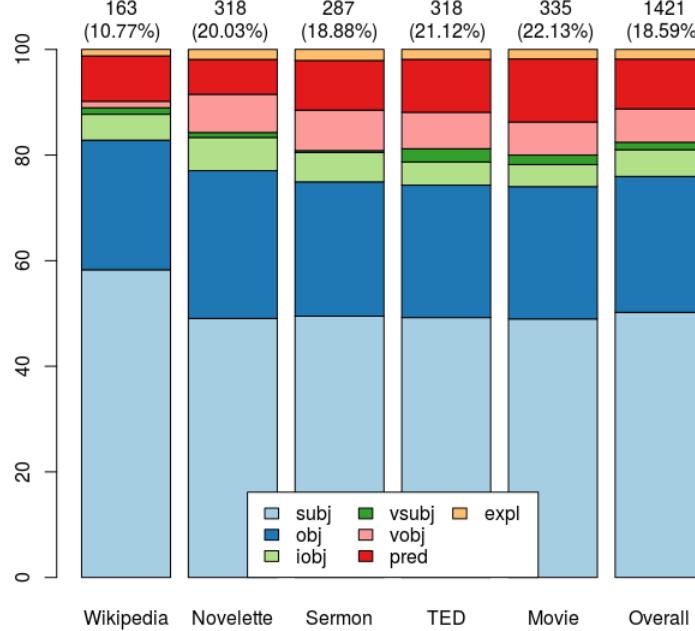


Figure 1: Distribution of dependency relations in the gold data. The numbers on top of the columns indicate the total frequencies of the seven relations considered in the evaluation and, in parentheses, the corresponding percentage (e.g. in the Wikipedia subcorpus, 163 tokens, which correspond to 10.77% of all tokens, have been annotated with one of the seven relations).

the systems described here.

Most prominently, there are those systems which provide a full range of annotations from tokenization to dependency parsing: CoreNLP (Manning et al., 2014), which combines rule-based tokenization with statistical tagging and dependency parsing, spaCy v2.1.3 (Honnibal and Montani, 2017),¹¹ and the Python-based StanfordNLP (Qi et al., 2018).

We also test a few dedicated tokenizers: NLTK (Bird et al., 2009), comprising a regular expression-based word tokenizer and the Punkt tokenizer (Kiss and Strunk, 2006) for sentence segmentation, SoMaJo (Proisl and Uhrig, 2016), and Syntok (Leitner, 2019). We evaluate both the rule-based sentencizer included in the spaCy pipeline as well as the sentence boundaries derived from spaCy’s dependency analysis, which is the usual way of determining sentence boundaries with spaCy. These results are listed under ‘spaCy parser’ in Table 4.

Some tools only perform lemmatization, such as IWNLP (Liebeck and Conrad, 2015), which makes use of a word form to lemma mapping extracted from the German Wiktionary, and GermaLemma (Konrad, 2019) (with the Pattern extensions¹² en-

abled), based on the TIGER corpus.

Since GermaLemma only lemmatizes words from a restricted set of POS (N*, V*, ADJ* and ADV*), i.e. content words, we combined this system with another tool that also lemmatizes function words (spaCy) and added a few special rules for pronouns. This ensemble lemmatizer is included in the study as GermaLemma++.¹³

Many of the tools we test provide some combination of word-level annotations. SoMeWeTa (Proisl, 2018) produces STTS tags (small tagset) only, whereas RFTagger (Schmid and Laws, 2008)¹⁴ and CleverTagger (Sennrich et al., 2013) use a modified version of the large STTS that includes morphological features. TreeTagger (Schmid, 1994; Schmid, 1995) provides just STTS and lemmas.

We also test ParZu (Sennrich et al., 2009), which is a parser that combines a hand-written grammar with statistical approaches.

Of the selected tools, RNNTagger (Schmid, 2019) and StanfordNLP (Qi et al., 2018) are both neural network-based systems implemented in the PyTorch framework. SpaCy uses convolutional

¹¹<https://github.com/explosion/spaCy/releases/tag/v2.1.3>

¹²Smedt and Daelemans (2012), <https://www.clips.uantwerpen.be/pages/pattern>.

¹³<https://github.com/rubcompling/germalemmaplusplus>

¹⁴Specifically the Java interface RFTJ (Ziai and Ott, 2014), which offers easier production of STTS tags, improved lemmatization, as well as finer control over model loading.

neural network models for tagging and parsing, but does not need a GPU or significant resources for annotation or training.

5 Evaluation

In this study we evaluate the NLP tools with respect to both accuracy and speed. Section 5.1 describes the performance evaluation, i.e. annotation speed of the selected systems. In Section 5.2, the comparison of the different annotations, as produced by the systems, with the gold standard annotation is explained.

5.1 Computational Efficiency

We are interested not only in the annotations that the systems we test produce, but also in the computational resources they require to produce them. This is particularly a concern in the context of practical applications where timely results are critical.

We wanted to measure the resources each system requires to produce each level of annotations, but the production of each type of annotation is not always separable from the others, e.g. morphology and/or lemma annotations are often produced together with POS tags (and though there are sometimes options regarding whether or not they are output, this doesn't necessarily mean that they aren't computed). Also, sentence segmentation and word tokenization are sometimes dependent on one another, but in different directions. CoreNLP requires sentence boundaries before tokenization, and others, like Syntok, do this the other way round.

For each annotation step, we separated model loading from actual annotation time required. We measure the time the various systems require to complete three roughly-comparable annotation steps: (1) tokenization (sentence segmentation, word tokenization), (2) word-level annotations (POS, morphology, lemmas), and (3) dependencies. Systems that only perform lemmatization are run separately, since they also require POS annotations as an input, all other systems produce lemmas and POS annotation simultaneously.

The annotation time for each step was measured as CPU time across five trials for each of the subcorpora, using a measure of seconds per thousand tokens, which represents the computational resources required by a particular system for a particular task. All trials were performed on a Linux workstation equipped with an Intel Core i7-5820K processor, 15 GB of RAM, and an Nvidia GeForce GTX 980

graphics card with 4 GB of memory.

5.2 Accuracy

For the accuracy evaluation, the systems were provided with gold annotations from the previous annotation steps, as detailed here:

	Input (Gold)	Output
Tokenization	Plain text	→ Sentences, Tokens
Word-level	Sentences, Tokens	→ POS, Morph, Lemmas
Lemmatization	POS	→ Lemmas
Dependencies	POS	→ Dependencies

Of course this approach is not a realistic scenario, as manually created, i.e. completely correct, annotations are normally not available, so these results are to be interpreted as an upper bound.

Tokenization In the Universal Dependencies Treebank for German,¹⁵ on which the StanfordNLP model was trained, multi-word tokens (APPRART in the STTS) are split into separate words. To enable the comparison of the system's output with the gold standard, we suppressed these splits during tokenization.

POS The deviating tokenization of multi-word tokens also affects the other annotation levels, as the corresponding models expect the tokens to be split for them to be annotated correctly. We therefore accepted split multi-word tokens for the concerned systems (StanfordNLP) for all word-level annotations and used rules to check if the system's annotation (APPR + ART) matches the gold-standard annotation (APPRART).

Morphology The morphology annotations of the systems follow different naming conventions for the morphological features (e.g. Sg vs. Sing for ‘singular’), so, in order to compare them, we mapped all annotations to the TIGER tagset (Crys-mann et al., 2005). Some systems also annotate further morphological features, which we ignored either because they are not present in the TIGER tagset (features like definiteness) or because they are already included in the POS tags (e.g. finiteness).

¹⁵UD German GSD (https://universaldependencies.org/treebanks/de_gsd/index.html)

Morphology tags are composed of multiple features, e.g. 3.P1.Pres.Ind, which we evaluate individually. Features that are only present in the system output are ignored. The overall accuracy is calculated token-wise as the average percentage of morphological features in the gold standard that were present and correct in the system output.

Lemmas The NLP tools included in this study follow different guidelines for the annotation of lemmas, so we allow for a degree of variation in this regard, so long as it is clear that the analysis a given lemma variant suggests is correct. For instance, most prominently, some use the masculine form of the definite article as its lemma, *dem* → *der*, whereas others might use the feminine form *die*. For words tagged as PPER, the surface form is accepted as its lemma. Reflexive pronouns may be lemmatized as *sich*, and words tagged as ART, PDS or PRELS may be lemmatized as *eine* as well as *ein* or *die* as well as *der*. We also consider <ß> and <ss> to be equivalent throughout.

Furthermore, we accept some special lemmas used by certain systems. For example, since RFTagger lemmatizes all numbers to a common symbol, either <card> or <ord>, we consider these symbols equivalent to the token’s surface form, since this is how we lemmatized numbers in our gold-standard dataset. Similarly, TreeTagger lemmatizes multi-word tokens tagged as APPPART with the lemmas of the constituent prepositions and articles, as in *zu+die* for *zur* instead of *zu*, as in our dataset. We also accept these forms as correct.

In general, all comparisons of lemmas are case-insensitive and wherever a tool does not produce a lemma for a given token, we take the surface form of that token as its lemma. If a tool outputs more than one alternative lemma for a word, as is the case for StanfordNLP, IWNLP, RFTagger and TreeTagger, we evaluate the first one.

Dependencies For the dependency evaluation, we only consider the relevant argument relations (subj, obj, iobj, vsubj, vobj, pred, expl). The four parsers compared in the study follow three different guidelines for dependency annotation and each of them uses a different scheme. In order to account for differing naming conventions, we accept a number of alternative labels for each relation, which we consider to be equivalent. To capture the structural differences between guidelines we use a set of rules to allow for certain mismatches between the sys-

tem outputs and the gold standard. In particular, the head of a relation may be the main verb, as in the gold standard, or an accompanying finite auxiliary as annotated by the parsers. Similarly, the head of a predicative phrase may be the noun, as is the case in the gold standard, or the preposition (for ParZu and spaCy). We also cover the copula analysis of CoreNLP and StanfordNLP, which according to the Universal Dependencies guidelines,¹⁶ analyse the predicative phrase as the head of a copula construction. Finally, there are certain constructions in German for which the subject is difficult to determine (also theoretically). These constructions involve certain predicatives and expletives (e.g. as in *Das ist es* ‘that’s it’). To address this, we consider it correct if the system switches the subject and the predicative phrase, and similarly, if the system analyses expletives as subjects (or objects, in rare cases) and vice versa. The StanfordNLP parser performs much better when Universal POS tags are present, so we add these to the data provided to this parser using the mapping provided by the Universal Dependencies project.¹⁷

6 Results and Discussion

In this section, we examine the most important results of our experiments. All of the scripts we used and detailed results (including per-register results) can be found in this paper’s repository at <https://github.com/rubcompling/konvens2019>.

6.1 Computational Efficiency

Figure 2 shows the results of run time evaluation. Overall, the annotation speed of most systems does not differ substantially. The word-level StanfordNLP component, encompassing POS tagging, morphological analysis, and lemmatization, is an outlier, with much longer run times, on average, than the other systems. The RNNTagger, another neural network-based system, is also on the upper end of the scale among the systems we tested. This is indicative of the large computational resources such systems require.

Though the CoreNLP dependency parser is the slowest of those we tested, CoreNLP is the fastest system for word-level annotations (which is probably related to the fact that it only produces POS

¹⁶<https://universaldependencies.org/guidelines.html>

¹⁷<https://universaldependencies.org/tagset-conversion/de-stts-uposf.html>

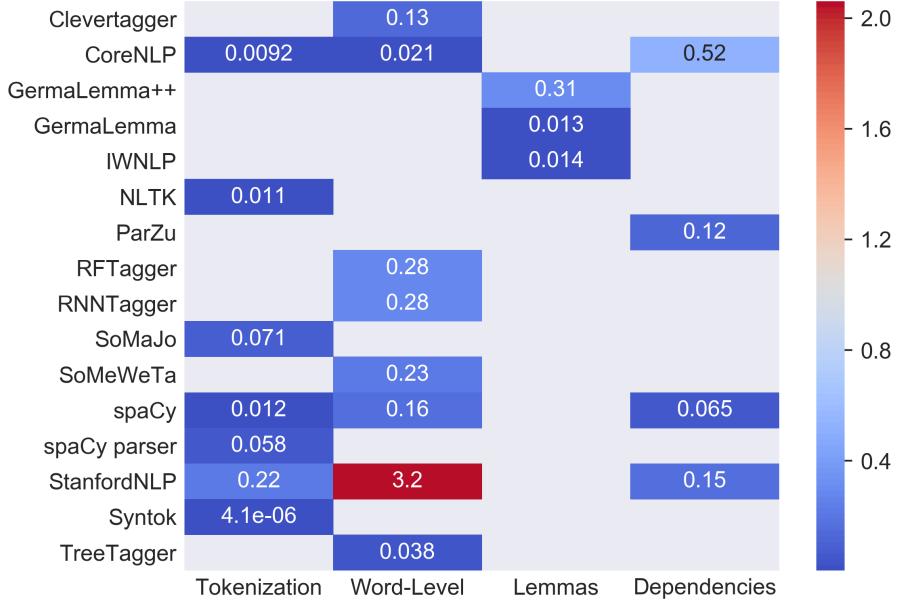


Figure 2: Process time required by the systems according to task, measured in seconds per thousand tokens.

annotations in this step), though this appears to come at the cost of accuracy. As was the case in Choi et al. (2015), spaCy is capable of the fastest dependency parsing.

The fastest system for tokenization is Syntok (another outlier) requiring only 0.000004 seconds per 1,000 tokens. It doesn't seem to use any linguistic models as such – just a few well-selected regex-based heuristics. Despite the minimal computational resources required, Syntok is still the second most accurate sentence segmenter we tested.

6.2 Accuracy

Table 4 shows the results of the accuracy evaluation for all systems and annotations.¹⁸

Tokenization For sentence segmentation and tokenization, we calculated F_1 -scores for all systems. The results show that sentence segmentation is more challenging than word tokenization which can be considered a solved task.

A closer look at the system output shows that many of the systems have difficulty recognizing abbreviations, ordinal numbers (e.g. in dates), ellipsis dots, and dashes that are used to indicate speaker changes in movie dialogues. Further problems are caused by direct speech, especially as regards the

use of inward-pointing angle quotes (as in »example text«) which are typical for German literary texts but appear to be unexpected for some of the systems. Also the data contains some sentences which are not marked with punctuation marks and can only be recognized as sentences based on the content.

POS The results for POS tagging are similar to the findings of Giesbrecht and Evert (2009) and Beißwenger et al. (2016) with tagging accuracies ranging from 88.2% to 94.3%. However, there are only slight differences between the selected registers. Most taggers perform best on the TED talk transcripts while the lowest accuracy values can be observed for movie subtitles.

The most frequent errors for all taggers include the common confusions of nouns and proper names, adverbs and adverbial adjectives, and of different verb forms. It is striking that all of the tools also tag sentence internal punctuation incorrectly, e.g. as numbers, names, adjectives, etc., which is the most frequent error for half of the taggers reducing their accuracy by up to 1.6 percentage points.

Morphology The accuracy values for morphology annotation differ substantially between systems with results varying by 10 percentage points.

Depending on their POS, words have different morphological features. Following the TIGER

¹⁸As the formulation of sensible baselines for several of the annotation steps, e.g. morphology and dependencies, can be quite complex, we don't include them in our analysis here.

	Tokens	Sents	POS	Morph	Lemmas	Deps
Clevertagger	–	–	0.8818	0.8338	–	–
CoreNLP	0.9965	0.8909	0.9074	–	–	0.5316
GermaLemma++	–	–	–	–	0.9541	–
GermaLemma	–	–	–	–	0.8786	–
IWNLP	–	–	–	–	0.8650	–
NLTK	0.9961	0.9177	–	–	–	–
ParZu	–	–	–	0.8359	0.9431	0.7744
RFTagger	–	–	0.9310	0.8903	0.9377	–
RNNTagger	–	–	0.9346	0.9248	0.9751	–
SoMaJo	0.9964	0.8480	–	–	–	–
SoMeWeTa	–	–	0.9403	–	–	–
spaCy	0.9955	0.8410	0.9250	–	0.8913	0.6687
spaCy parser	–	0.8940	–	–	–	–
StanfordNLP	0.9920	0.8989	0.9427	0.8235	0.9415	0.7217
Syntok	0.9912	0.9125	–	–	–	–
TreeTagger	–	–	0.9210	–	0.9605	–

Table 4: Overall F_1 -scores (for tokens and sentences) or accuracy (for POS, morphology, lemmas and dependencies) for all systems at all annotation levels.

scheme (Crysmann et al., 2005), we evaluate seven features: gender, case, person, number, tense, mood, and degree. Table 5 provides detailed results for all features.

As morphology annotation depends on correct POS tags, it is possible that errors on this annotation level are the result of error propagation from the POS level, e.g. adjectives that are tagged as adverbs do not receive a degree annotation, finite verbs that are recognized as infinitives are not annotated at all. Sometimes the systems also indicate that they cannot assign a unique value to an attribute although this is possible for human annotators based on the given context.

While no clear differences between registers can be observed, there are some morphological features that generally seem to be harder than others. Overall, the most difficult features are gender and case with error rates of 10% or more for all systems. The annotation of gender is most difficult for proper names and pronouns, while error rates are low for nouns and articles.

It should also be noted that StanfordNLP does not annotate the degree feature for adjectives, which occurs with 10% of the annotated tokens and makes up 3% of the annotated features overall. This explains, in part, the system’s poor result in this annotation task.

Lemmas Most of the lemmatizers achieve accuracy values well above 90%. Always choosing a given word form as its lemma results in an accuracy of 70.7%. In general, there are no significant differences in lemmatization between registers. However, all of the systems do perform slightly worse on the Novelette subcorpus than on the other subcorpora.

We observe frequent deviations from the gold standard for demonstrative and indefinite pronouns, which can likely be attributed to further differences between lemmatization guidelines. Some systems also produce stems instead of lemmas for some words, e.g. StanfordNLP and TreeTagger annotate the stem *jen* instead of possible lemmas *jener* or *jene*. Furthermore, some systems do not lemmatize certain words or word classes at all, for instance content words in the case of GermaLemma, which results in a lower accuracy as well.

Dependencies The evaluation shows that dependency annotation is clearly the most difficult of the six annotation tasks. All four parsers we tested achieve accuracies for the annotation of arguments below 78%. A closer look at the results shows that the most difficult relations seem to be clausal subjects, clausal objects and expletives.

The spaCy parser produces the largest number of false positives, i.e. argument relations that are not present in the gold standard (mostly object clauses).

	Case	Degree	Gender	Mood	Number	Person	Tense
Clevertagger	0.8385	0.9217	0.6537	0.9009	0.9261	0.9108	0.8803
ParZu	0.8104	0.8733	0.7913	0.6744	0.8870	0.9388	0.9743
RFTagger	0.8858	0.9447	0.7734	0.9537	0.9522	0.9492	0.9550
RNNTagger	0.9094	0.9401	0.8770	0.9730	0.9547	0.9713	0.9717
StanfordNLP	0.8563	–	0.7634	0.9189	0.9527	0.8954	0.9408

Table 5: Overall accuracy per morphological feature.

This may in part result from differences in handling coordination: While spaCy tends to annotate the affected relation once per coordinated element, other guidelines use a special coordination relation for this.

CoreNLP has the second highest number of false positives. It also makes the most mistakes of the four parsers and has the highest error rates for almost all relations, resulting in an accuracy value only just above 50%.

StanfordNLP and ParZu both reach accuracies above 70%. ParZu achieves the best result of all parsers overall and produces much fewer false positives.

However, it often annotates multiple roots per sentence and/or annotates subjects, objects or clausal objects as root of the sentence. It also seems to have a slightly different definition of predicates and annotates expletives worse than the other systems (although no system annotates expletives particularly well).

7 Conclusions and Future Work

Though we would like to directly relate accuracies to run times and provide some concrete measure of efficiency, this is difficult in practice, since run times pertain to multiple accuracy dimensions: the run time of the POS annotation step sometimes covers just POS, morphological annotations, and/or lemmatization, depending on what a given system provides, which makes it difficult to say what the relationship is between run time and accuracy exactly. Nevertheless, we will attempt to take the systems’ run times into account when comparing the accuracies achieved.

Though the F_1 -scores above 0.99 for all tested tokenizers suggest that tokenization can be considered a solved task, there is some degree of difference in how the tools handle sentence segmentation. Overall, NLTK provides some of the best token and sentence boundaries, while being one of the fastest

systems we tested. Syntok is even faster and only slightly less accurate.

RNNTagger offers high accuracy across all word-level annotations and is relatively fast, especially as compared with taggers offering similar levels of accuracy. Though the StanfordNLP tagger seems to be a bit more accurate at POS tagging, it does so at the cost of requiring much greater computational resources. If you have POS annotations and just need lemmas, then GermaLemma++ provides the best accuracy with only a modest increase in the computing resources required, in comparison to standard GermaLemma.

SpaCy offers relatively good performance, but requires especially little computational resources and can annotate large volumes of data quickly. However, where accuracy is most important, ParZu might provide better results while still being relatively fast.

Future work could include retraining the systems for each domain and observing the degree to which these domain-specific models improve the accuracy of the systems’ output. Furthermore, one could use these results to construct a pipeline from the appropriate systems for a particular situation and have an easy-to-use and effective NLP pipeline for the standard linguistic annotations. It would also be interesting to evaluate in future work how well such a pipeline (in which each stage depends on system output, as opposed to gold-standard annotations) would compare to the upper bound results described here.

Acknowledgments

We would like to thank our student annotators Doreen Scholz, Larissa Weber, Jennifer Wodrich, Helena Wedig, Sara Klein, Anna Ehlert and Katharina Streit for the annotations. This work was supported by the German Research Foundation (DFG), SFB/CRC 1102 “Information density and linguistic encoding” (Project C6).

References

- Michael Beßwenger, Sabine Bartsch, Stefan Evert, and Kay-Michael Würzner. 2016. EmpiriST 2015: A shared task on the automatic linguistic annotation of computer-mediated communication and web corpora. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X) and the EmpiriST Shared Task*, pages 44–56.
- Douglas Biber and Susan Conrad. 2009. *Register, Genre, and Style*. Cambridge University Press.
- Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.
- Jinho D. Choi, Joel Tetreault, and Amanda Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 387–396.
- Berthold Crysmann, Silvia Hansen-Schirra, George Smith, and Dorothea Ziegler-Eisele. 2005. *TIGER Morphologie-Annotationsschema*. Retrieved from https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/TIGERCorpus/annotation/tiger_scheme-morph.pdf.
- Richard Eckart de Castilho, Éva Mújdricz-Maydt, Seid Muhie Yimam, Silvana Hartmann, Iryna Gurevych, Anette Frank, and Chris Biemann. 2016. A web-based tool for the integrated annotation of semantic and syntactic structures. In *Proceedings of the workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH) at COLING 2016*, pages 76–84.
- Phani Gadde, L.V. Subramaniam, and Tanveer Faruquie. 2011. Adapting a WSJ trained part-of-speech tagger to noisy text: Preliminary results. In *Proceedings of the 2011 Joint Workshop on Multilingual OCR and Analytics for Noisy Unstructured Text Data*.
- Eugenie Giesbrecht and Stefan Evert. 2009. Part-of-speech tagging – a solved task? An evaluation of POS taggers for the web as corpus. In *Proceedings of the 5th Web as Corpus Workshop (WAC5)*, pages 27–35.
- Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: short papers*, pages 42–47.
- Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.
- Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.
- Markus Konrad. 2019. GermaLemma: A lemmatizer for German language text. <https://github.com/WZBSocialScienceCenter/germalemma>.
- Florian Leitner. 2019. Syntok: Text tokenization and sentence segmentation (segtok v2). <https://github.com/fnl/syntok>.
- Matthias Liebeck and Stefan Conrad. 2015. IWNLP: Inverse Wiktionary for natural language processing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 414–418, Beijing, China.
- Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.
- Melanie Neunerdt, Michael Reyer, and Rudolf Mathar. 2013. A POS tagger for social media texts trained on web comments. *Polibits*, 48:61–68.
- Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, and Nathan Schneider. 2012. Part-of-speech tagging for Twitter: Word clusters and other advances. Technical report, School of Computer Science, Carnegie Mellon University.
- Thomas Proisl and Peter Uhrig. 2016. SoMaJo: State-of-the-art tokenization for German web and social media texts. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X) and the EmpiriST Shared Task*, pages 57–62, Berlin, Germany. Association for Computational Linguistics (ACL).
- Thomas Proisl. 2018. SoMeWeTa: A part-of-speech tagger for German social media and web texts. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 665–670, Miyazaki, Japan. European Language Resources Association ELRA.
- Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium, October. Association for Computational Linguistics.

Anne Schiller, Simone Teufel, Christine Stöckert, and Christine Thielen. 1999. *Guidelines für das Tagging deutscher Textcorpora mit STTS (Kleines und großes Tagset)*. Retrieved from <http://www.sfs.uni-tuebingen.de/resources/stts-1999.pdf>.

Helmut Schmid and Florian Laws. 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 777–784, Manchester, UK.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, Manchester, UK.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to german. In *Proceedings of the ACL SIGDAT Workshop*, Dublin, Ireland.

Helmut Schmid. 2019. RNNTagger – a neural part-of-speech tagger. <https://www.cis.uni-muenchen.de/~schmid/tools/RNNTagger/>.

Rico Sennrich, Gerold Schneider, Martin Volk, and Martin Warin. 2009. A new hybrid dependency parser for German. In *Proceedings of the GSCL Conference*, Potsdam, Germany.

Rico Sennrich, Martin Volk, and Gerold Schneider. 2013. Exploiting synergies between open resources for German dependency parsing, POS-tagging, and morphological analysis. In *Proceedings of Recent Advances in Natural Language Processing*, pages 601–609, Hissar, Bulgaria.

Tom De Smedt and Walter Daelemans. 2012. Pattern for Python. *Journal of Machine Learning Research*, 13:2063–2067.

Ramon Ziai and Niels Ott. 2014. RFTagger Java interface. <http://sifnos.sfs.uni-tuebingen.de/resource/A4/rftj/>.

Towards a gold standard corpus

for detecting valencies of Zulu verbs

Gertrud Faaß

Hildesheim University

Institute for Information Science

and Natural Language Processing

and University of South Africa

Department of African Languages

gertrud.faaß@uni-hil-
desheim.de

Sonja Bosch

University of South Africa

Department of African Languages

boschse@unisa.ac.za

Abstract

We report on a new project building a Natural Language Processing resource for Zulu by making use of resources already available. Combining tagging results with the results of morphological analysis semi-automatically, we expect to reduce the amount of manual work when generating a finely-grained gold standard corpus usable for training a tagger. From the tagged corpus, we plan to extract verb-argument pairs with the aim of compiling a verb valency lexicon for Zulu.

1 Introduction

The observation that all parts of speech in a phrase, clause or sentence interact in some way with each other is one of the most important basics of today's grammars. With regard to Head-Driven Phrase Structure Grammar (HPSG, cf. Pollard and Sag, 1994), Sag et al. (2003:536) state that 'all the parts of a phrase depend directly on its head word'. Looking at the constraint-based Lexical Functional Grammar (LFG, cf. Bresnan, 2001, cited by Butt et al., 1999:43), we note that the 'determination of a verb's subcategorization frame [...] constitutes a central part of any grammar development effort'. In accordance with the perspective of (lexical) semantics, a verb that for instance denotes a change of state requires one or more 'participants', the *arguments* of the verb that will represent the actor, and/or the thing or person experiencing the change-of-state described by this verb.

As the type and number of arguments of a verb depend on its use, the availability of large text collections (i.e. corpora) is essential when attempting to generate a lexicon of verb valencies. With regard to an under-resourced language such as Zulu, a relatively large corpus has only been made available recently, hence we can start working towards generating a verb valency lexicon for Zulu, combining known methods and available tools with the aim of a - at least mostly automated – processing chain.

2 Zulu challenges

Zulu is a member of the Bantu language family and is one of the eleven official languages of South Africa. The morphological structure of Zulu is depicted by a nominal classification system according to which nouns have prefixal morphemes (so-called noun prefixes). For ease of reference these noun prefixes have been assigned numbers by scholars working in the field of Bantu linguistics. The various noun prefixes link the noun to other words in the sentence, e.g. verbs, adjectives, possessives, pronouns, and so forth by means of concordial morphemes or concords. Zulu is predominantly agglutinating in nature, with the majority of words consisting of more than one morpheme which is as such, a challenge for NLP processing. Like other languages with a highly informative morphology, Zulu also allows for a relatively free word order (cf. Gowlett, 2003:636).

While examining any Zulu grammar book, readers are often surprised by the hundreds of forms a verb, for instance, can appear in. Ten different tempi can be distinguished. Polarity and modality are encoded in the verb, too. Zulu is not content

with a first, a second and a third person in singular and plural: the third person is split into sixteen *noun classes* (of which two have sub-classes, and about half express the singular while the others stand for plural and abstract forms). In order to express subject-verb congruence a *subject concord* exists for each noun class, as shown in Table 1.

<i>word form</i>	<i>analysis</i>	<i>Translation</i>
<i>ngihamba</i>	<i>ngi_{1ps-sg}-hamb-a</i>	I walk
<i>uhamba</i>	<i>u_{2ps-sg}-hamb-a</i>	you walk
<i>uhamba</i>	<i>u_{cl1-sg}-hamb-a</i>	he/she/it walks
...		
<i>lihamba</i>	<i>li_{cl5-sg}-hamb-a</i>	he/she/it walks
<i>ahamba</i>	<i>a_{cl6-pl}-hamb-a</i>	they walk
...		

Table 1. *hamba* ("walk"): Partial inflection paradigm of the present tense indicative

Object concords may also appear as part of the orthographic verb and they may either co-occur or substitute an overt object in the sentence. As a demonstration of the latter phenomenon, the orthographic verb *bayakupheka* ("they are cooking it (at the moment)") that actually expresses an entire clause, is explained from a morphological perspective in Table 2.

morph	<i>ba-</i>	<i>-ya-</i>	<i>-ku-</i>	<i>-phek-</i>	<i>-a</i>
categ.	subj.	pres.	obj.	verb root	verb ending
	con-	ind.	con-		
	cord	long	cord		
	cl. 2	form	cl.		
Engl.	they	now	it	cook	
transl.	they are cooking it (at the moment).				

Table 2. Analysis of *bayakupheka* ("they are cooking it (at the moment)")

Lastly, one may find suffixes in verbs that modify their valency. These suffixes have meanings similar to prepositions in languages like English or German and, just like these, they require arguments. Adding the applicative suffix *-el* for example to a verb changes its valency: it now needs an additional argument describing a beneficent. This issue is demonstrated in Table 3 for the

verb form *ngipheka* ("I cook") becoming *ngiphekela* ("I cook for"). Such a derivation often changes the meaning of the verb as well (cf. Bosch and Pretorius, 2017).

<i>word form</i>	<i>analysis</i>	<i>transl.</i>
<i>ngipheka</i>	<i>ngi_{1ps-sg}-phek-a</i>	I cook
<i>ngiphekela</i>	<i>ngi_{1ps-sg}-phek-el_{appl-a}</i>	I cook for

Table 3. Application of the applicative suffix *-el*

3 Aims and Resources

Our long-term aim is to compile a corpus-based machine-readable valency lexicon for Zulu verbs which will be freely available for research purposes. By generating this lexicon, we expect to be able to explore the syntax of the Zulu language in use on a bigger scale than previously possible.

However, there is still a long way to go: thus far, Zulu text taggers (Spiegler et al., 2010; Koleva, 2011; De Pauw, 2012; Eiselen and Puttkammer, 2014) are all using a rather coarse tagset not applicable for our purposes. Second, except the tagger by Eiselen and Puttkammer (2014) none of these taggers seems to be available for local use¹.

In summary, the following list shows our primary short term aims:

1. developing a more informative tagset,
2. generating a gold standard corpus fully annotated with the tagset,
3. training and evaluating taggers and tagset, and
4. developing a chunker for extracting verbs and their arguments.

This paper is concerned with the first two steps, and it is describing the corpus and the tagging processes that have been done so far.

In the last decade, a number of NLP resources for Zulu were compiled. Most of them are available at the *South African Centre for Digital Language Resources* (SADILaR)², inter alia a Zulu Tagger (Eiselen and Puttkammer, 2014). This tagger is listed as the NCHLT Tagger.

Another important resource for our project is the 3-million token Zulu corpus compiled in the *Wortschatz* project in Leipzig³. This corpus has been extended recently to 15.4 million tokens

¹ De Pauw's (2012) tagger can be applied online via (<https://www.aflat.org/zulutag>).

² <https://www.sadilar.org/>

³ Leipzig Corpora Collection (2016): *zul_mixed_2016* based on texts of the year 2016. Leipzig Corpora Collection.

which will also be made available for free download. Lastly, we make use of the ZulMorph morphological analyser available as a Finite state morphology demo and reported on in detail in several publications, e.g. Bosch and Pretorius (2011). An attempt was also made to get the other taggers described above (Spiegler et al., 2010; Koleva, 2011; De Pauw, 2012) for local use, although their tagsets are not very useful. However, our requests to the authors of the respective papers were not successful.

4 Application of resources

4.1 Corpus

The currently available Zulu corpus of the Leipzig Wortschatz Collection contains more than 3 million tokens with marked sentence borders. We selected 149,196 sentences (2,337,566 tokens) in total for our local processing after deleting noise. To build our gold standard corpus, we randomly selected 1,500 sentences (about 17k tokens) from this resource.

4.2 Tagset and Tagger

The Zulu tagset used by the NCHLT Tagger (Eiselen and Puttkammer, 2014), includes nouns (*Nn*), adjectives (*An*), and verbs (*V*) of which the former two have noun classes (*n*) assigned, e.g. N01 or A07.

Because of the agglutinative orthography of the Zulu language, a number of syntactic constructions like copularives (COP) and possessives (POS*n*) have their own tags assigned. This issue was criticized by e.g. Hendrikse and Mfusi already in 2008, calling for a tagset that marks such constructions as clauses, a suggestion that we will implement.

As to pronouns, there are tags for personal pronouns (PRON*n*), demonstratives (DEM*n*), and quantitatives (QUANT*n*). Unfortunately, the tagger also assigns tags like "PRON", "QUANT" or "REL" ("relative") without naming a noun class and we even find an undescribed tag "P" (we assume that this stands for "any kind of pronoun").

There are tags labelling ideophones (IDEO), though these either function as adverbs or as verbs in a sentence. We also find tags for adverbs (ADV), numeratives (NUM), conjunctions (CONJ), and interjections (INT).

The NCHLT Tagger moreover makes use of the tag "M" for which we do not find any description in the NCHLT Project⁴. The tag labels a variety of items, like (copulative) verbs but also proper nouns and abbreviations.

When developing their tagger, Spiegler et al. (2010) collapsed the two noun classes 8 and 10 into one as their forms are identical. Before tagging, they also deleted punctuation in the text and changed all characters to lower case thus their corpus is not in its original form any longer. Other developers (Koleva, 2011; De Pauw, 2012) based their works on the tagset of Spiegler et al. (2010), but they did not differentiate between noun classes at all (therefore gaining a high precision).

As described above, the NCHLT Tagger, Eiselen and Puttkammer (2014) make use of a tagset that distinguishes noun classes, however all verbs are labelled with the tag "V", which means that a subject-verb congruence cannot be detected.

In a first go, we utilize the NCHLT Tagger as it is the only tagger than can be applied to our corpus (and that seems to be available), however we need to extend the tag "V" with subject and object class information whenever this information is available and we must train a new tagger, as the NCHLT Tagger comes without the possibility to adapt it to other tagsets. We must also be aware of the fact that this tagger has not been evaluated and that it applies tags "P" and "M" which we both define as "miscellaneous" categories not usable for further processing of tagged text.

4.3 ZulMorph

The ZulMorph morphological analyser (Bosch and Pretorius, 2006) is unfortunately not available for offline use, but the developers process lists of words on request. Currently, there are about 36,000 verb roots described (Pretorius and Bosch, 2017) in ZulMorph. When applying it to our 8,625 types (see 5.2), 1,895 types were not analysed.

4.4 Combining information provided by the tools

The need for a finer-grained tagset and a procedure allowing us to generate a gold standard leads us to an idea described in Jung's dissertation (ne Eckart, 2018). Jung suggests the combination of information provided by different tools in order to achieve a better result.

Dataset. https://corpora.uni-leipzig.de/de?corpusId=zul_mixed_2016

⁴ <https://repo.sadilar.org/handle/20.500.12185/351>

We hence plan to apply a “voting” procedure: in case the NCHLT Tagger agrees with the ZulMorph analysis for closed class items (like CONJ), we will not check the results again, if the NCHLT Tagger votes for V while ZulMorph offers V and non-V analyses, we will choose the V-analyses. If there are several, a semi-automatic selection of the correct analysis will take place.

5 Intermediate Results

5.1 Tagset

Our preliminary tagset is built on four levels, of which the first two are shown in Table 4. The first level describes the coarse category (to allow for future coarse tagging), the second level describes the part-of-speech in more detail. For verbs, we distinguish regular finite forms and forms with suffixes modifying their valency (NEUT(er), APPL(icative), RECIP(rocal), CAUS(ative) and PASS(ive)).

For nominal items, a third level specifying the noun class will be utilized. For verbs, this third level contains the noun class of the subject noun, the fourth level then describes (if available), the noun class of its object in the cases where an object concord appears. This fourth level is filled with the letters “RF” in case the verb contains a reflexive prefix. The tagset does not distinguish positive from negative polarity for this factor does not change its valency.

<i>1st level</i>	<i>2nd level</i>	<i>Description</i>
V(erb)	APPL	applicative
	CAUS	causative
	COP	copulative (x is V)
	IMP	imperative
	IDEO	ideophone
	FIN	finite (inflected form)
	NEUT	neuter
	RECIP	reciprocal
	PASS	passive
	RELP	verb containing a relative clause
N(oun)	COPP	nominal copulative clause (N is N)
	INF	infinitive (noun prefix and verbal stem)
	PROP	proper noun
	POSP	noun containing a possessive clause

	REG RELP	regular noun noun containing a relative clause
ADJ(ec-tive)	COPP	adjectival copulative clause (x is A)
	REG	regular
ADV(erb)	IDEO	ideophone
	LOC	locative
	REG	true adverb
P(ronoun)	DEM	demonstrative
	PER	personal
	QUANT	quantitative
CONJ		conjunction
INTJ		interjection
INTR		interrogative
PUNCT		punctuation
CARD		anything containing numbers
FM		foreign language material

Table 4. Preliminary Zulu-tagset

5.2 Tagging

To gain tags from morphological analyses, we first extracted all 8,625 types of our corpus (note that in this number, upper- and lower-case forms were merged) and ran them through the ZulMorph tool. This resulted in 40,458 analyses, as there are types resulting in around 100 analyses in total (e.g. *abazi*, a word with several meanings⁵ that resulted in 105 different analyses).

The ZulMorph analyses contain a number of items not relevant for our purposes, we hence simplify those analyses reducing the amount of information provided. To gain a better overview, analyses like (1) of the word *omfundisayo* (“who teaches him/her”) are reduced to the relevant information, as in (2).

- (1) omfundisayo [RC][1]mu[OC][1]
fund[VRoot]is[CausExt]a[VT]yo[RelSuf]⁶
- (2) omfundisayo
o[RC][1][OC][1][CausExt][RelSuf]

From there, we can identify the verbal relative phrase, of which the subject is of noun class 1 and the object is of noun class 1 (tag: V-RELP-S01-O01). Note that for the word *omfundisayo*, there are altogether 6 ZulMorph analyses which our

⁵ The interested reader is referred to <https://isizulu.net/> for the variety of meanings of the word *abazi*

⁶ In this ZulMorph analysis, all processing information (unrelated to morphemes) was deleted.

scripts reduce to three: V-RELP-S01-O01 (subject identified as of noun class 1), V-RELP-S02ps-O01 (subject identified as 2nd Person Singular) and V-RELP-S03-O01 (subject identified as of noun class 3). For the word *ungomunye* (“you/he/she are/is the other one”), we find 45 ZulMorph analyses that are collapsed by our tool to 6 possible annotations, and for the above mentioned *abazi*, our implementation reduces the ZulMorph analyses from 105 to 13 possible tags.

However, there is still a need for a human expert to decide upon which of the found analyses is correct in the given context.

The scripts and tools developed so far select most of the analyses generated by ZulMorph fully automatically. There are currently still 3,297 types in our gold standard corpus to be identified. For these, we follow the following actions:

1. Collapsing further ZulMorph analyses to tags that can be annotated automatically;
2. identifying the possible tags of the types for which no ZulMorph analyses are available.

We currently assist the experts working on 2. with an automated detection of possible POS-tags by looking at orthography patterns of types. For example, names usually begin with a lowercase nominal prefix “u” or “i”, followed by the name beginning with an uppercase letter (e.g. uSiwela). We can annotate such types as N-PROP-01a automatically in order to avoid the necessity of human intervention.

6 Conclusion and future work

In conclusion, this project on the preparation of a future gold standard corpus for detecting valencies of Zulu verbs is still in its initial stages. So far, we have developed an informative tagset and found a methodology that makes use of available resources like the NCHLT Tagger and ZulMorph to assist us in assigning possible tags for each word of this corpus. This paper serves to describe our path towards achieving our goals and to elicit constructive feedback.

Our next steps entail the selection of the correct POS-Tag as soon as all possible POS-tags have been found for all types occurring in the training corpus, i.e. the future gold standard. In most cases, this selection must be done manually by language experts. As soon as the training corpus is finalized, we will make this corpus and a

full description of the preliminary tagset available via the SADiLaR repository.

After completing the gold standard corpus, we will train statistical taggers, evaluate the tagset and find the tagger best suited for the task. With this tagger, we will then annotate a new 20-million token Zulu corpus which will be provided by the University of Leipzig in pursuit of our goal of detecting verbs and their arguments on a bigger scale. We plan to also annotate only the first level of the tags in the course of the validation expecting a higher precision. We will also make the resulting annotated corpus available for other researchers who do not need a finer tagset for their purposes. The next goal is the development of a chunker for the identification of relevant verbal phrases from the corpus. After the phrase annotations have been added to the corpus, we will be able to generate the planned lexicon of verb valencies.

References

- Bosch S.E. and Pretorius L. 2006. A Finite-State Approach to Linguistic Constraints in Zulu Morphological Analysis. *Studia Orientalia*, 103, pp 205-227.
- Bosch S.E. and Pretorius L. 2011. Towards Zulu corpus clean-up, lexicon development and corpus annotation by means of computational morphological analysis. *South African Journal of African Languages*, 31(1), pp 138-158.
- Bosch S.E. and Pretorius L. 2017. A Computational Approach to Zulu Verb Morphology within the Context of Lexical Semantics. *Lexikos* (AFRILEX-reeks/series 27: 2017), pp. 152-182.
- Bresnan J. 2001. *Lexical-Functional Syntax*. Blackwell Publishing, Malden, USA, Oxford, UK, Victoria, Australia.
- Butt M., Holloway King T., Niño M.E. and Segond F. 1999. Automatic extraction of subcategorization from corpora. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington DC, USA.
- De Pauw, G., De Schryver, G-M. and van de Loo, J. 2012. Resource-Light Bantu Part-of-Speech Tagging. *Proceedings of the Workshop on Language Technology for Normalisation of Less-Resourced Languages* (SALT MIL8/Af-LaT2012). Istanbul: European Language Resources Association. pp 55-92.

- Eiselen E.R. and Puttkammer M.J. 2014. Developing text resources for ten South African languages. (In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, Reykjavik, Iceland. pp 3698-3703.
- Gowlett, D. 2003. Zone S. In D. Nurse and G. Philippson (eds): *The Bantu languages*, pp. 609-38. Routledge: London & New York.
- Hendrikse R. and Mfusi M. 2008. A morphosyntactic tagset for Southern Bantu within a Construction Grammar Approach, *Language Matters* (39:2), pp 181-203.
- Jung (ne Eckart), K. 2018. Dissertation: *Task-based parser output combination: workflow and infrastructure*. Universität Stuttgart: Fakultät Informatik. doi:10.18419/opus-9853.
- Koleva, M. 2011. M.A. Dissertation: *Towards Adaptation of NLP Tools for Closely-Related Bantu Languages: Building a Part-of-Speech Tagger for Zulu*. Saarbrücken: Universität des Saarlandes.
- Pollard C. and Sag I.A. 1994. *Head Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. The University of Chicago Press, USA.
- Pretorius, L. and Bosch, S. E. 2003. Finite-State Computational Morphology: An Analyzer Prototype for Zulu. *Machine Translation*, 18(3), pp. 195-216.
- Sag, I.A., Wasow, T. and Bender, E.M. 2003. *Syntactic Theory*. CSLI Lecture Notes Number 152, Stanford, California, USA, 2nd Edition.
- Spiegler, S., van der Spuy, A. and Flach, P. A. 2010. Ukwabelana - An open-source morphological Zulu corpus. *Proceedings of the 23rd International Conference on Computational Linguistics* (Coling 2010), 28. Beijing, China. pp. 1020-1028.

“Konservenglück in Tiefkühl-Town” – Das Songkorpus als empirische Ressource interdisziplinärer Erforschung deutschsprachiger Poptexte

Roman Schneider

Justus-Liebig-Universität

Angewandte Sprachwissenschaft und Computerlinguistik

Otto-Behaghel-Str. 10 D, 35394 Gießen

roman.schneider@germanistik.uni-giessen.de

Abstract

Der Beitrag beschreibt ein mehrfach annotiertes Korpus deutschsprachiger Songtexte als Datenbasis für interdisziplinäre Untersuchungsszenarien. Die Ressource erlaubt empirisch begründete Analysen sprachlicher Phänomene, systemisch-struktureller Wechselbeziehungen und Tendenzen in den Texten moderner Popmusik. Vorgestellt werden Design und Annotationen des in thematische und autorenspezifische Archive stratifizierten Korpus sowie deskriptive Statistiken am Beispiel des Udo-Lindenberg-Archivs.

1 Einleitung

Natürlichsprachliche Korpora als systematisch zusammengestellte Digitalisate von Kommunikationsakten bilden die wichtigste empirische Grundlage linguistisch motivierter Forschung. Für die standardnahe deutsche Gegenwartssprache existieren umfangreiche Korpusammlungen literarischer, journalistischer, juristischer, wissenschaftlicher und anderer weit verbreiteter Textsorten, ergänzt durch diverse Spezialcorpora zur Abdeckung spezifischer Sprachumstände (vgl. Kupietz/Schmidt 2018, Lemnitzer/Zinsmeister 2015, Lüdeling/Kytö 2008).

Bemerkenswert erscheint vor diesem Hintergrund das Fehlen einer wissenschaftlich validen, nachhaltig nutzbaren digitalen Sammlung von Popmusiktexten. So wie sich die Popmusik von einem ursprünglich jugendkulturellen Phänomen in den 1950er-/1960er-Jahren zu einem festen Bestandteil der Alltagskultur entwickelt hat, sind deren textuellen Inhalte in der Sprachrealität inzwischen allgegenwärtig und zunehmend Gegenstand (qualitativer) Forschung (vgl. von Ammon/von Petersdorff 2019). Wir sind von ihnen umgeben,

nicht nur beim Radiohören während des Autofahrens, beim Einkaufen im Supermarkt, via Online-Streamingdienst oder in TV-Shows. Hinzu kommt ein durchaus lyrischer Anspruch: Moderne Popsongtexte als „Gebrauchslyrik“ (Blühdorn 2003) sind „latent poetisch, aber selten authentisch poetisch“ (Flender/Rauhe 1989). Sie dienen oft nicht allein der simplen Zerstreuung, sondern werden genutzt, um Botschaften und Gefühle zu vermitteln oder – auf Rezipientenseite – Inspiration und Erklärungen zu finden.

Angesichts dieses beachtlichen „kommunikativen Impact Factors“ (Kreyer/Mukherjee 2007) besteht ein substanzielles Desiderat hinsichtlich der Berücksichtigung des Popmusik-Genres in der Korpuslinguistik. Keine der etablierten Sammlungen enthält Songtexte, entsprechend wenig erforscht sind spezifische Aspekte wie Ästhetik und Stil (Vokabular, Syntax, Register etc.), Inhalt (Thematiken, z. B. im historischen/politischen Kontext), Emotionalität (Kategorisierung, Intensität und Verteilung) oder Beziehungen zwischen Form und Inhalt. Wie für wenig erforschte Sprachgenres üblich, erscheinen initiale Erprobung und Validierung statistischer Maße und Verfahren aufschlussreich, auch hier stößt das Songkorpus in eine bestehende Lücke.

2 Stand der Kunst

Nachhaltige, empirisch begründete Forschung zu Texten deutschsprachiger Popmusik bleibt bislang aufgrund der Nichtexistenz ausreichend stratifizierter und aufbereiteter Daten ein unerfülltes interdisziplinäres Desiderat. Für das Englische hingegen lassen sich inspirierende Beispiele korpuslinguistischer Forschung zu Diskurs und Sprache in Songtexten finden. So enthält das BLUR-Korpus (Blues Lyrics Collected at the University of Regensburg; Miethaner 2005) mehr als 8.000 digitalisierte Texte und bildet damit eine wert-

volle Ressource für die Erforschung amerikanischer Bluessongs. Einen weiteren Meilenstein der Songtextforschung liefern Kreyer/Mukherjee (2007) mit dem von ihnen kompilierten Gießen-Bonn Corpus of Popular Music (GBoP), das englischsprachige Texte von Top-30-Alben empirisch auswertbar macht. Katznelson et al. (2010) und Cullem (2009) beschreiben Korpusanalysen zu amerikanischen Songtexten; Watanabe (2018) begründet das American Popular Music Corpus of English (PMCE-US). Bertin-Mahieux et al. (2011) haben ein „Million-Song-Dataset“ aufgebaut, während Murphey (1992) eine frühe Sammlung aus Top-50-Chartsongs kompiliert, quantitativ analysiert (z. B. hinsichtlich des Type-Token-Verhältnisses) und qualitativ auswertet (z. B. hinsichtlich der Verwendung von Pronomina). Weitere englischsprachliche Korpora existieren zu spezifischen Subdomänen, beispielsweise das Rock Lyrics Corpus (ROLC; Falk 2013).

Werner (2012) vergleicht amerikanisches und britisches Englisch in Popsongs und beschreibt Nutzungsaspekte für das Zweitsprachenlernen (Werner/Lehl 2015). Bereits Plitsch (1997) thematisiert den motivierenden Einsatz von Popmusiktexten für den Sprachunterricht, während Terhune (1997) hier insbesondere den syntaktisch oft nicht standardkonformen Aufbau von Songtexten kritisch sieht. Viol (2000) diskutiert identitätsstiftende Phänomene in britischen Popmusiktexten, Motschenbacher (2016) und Van Hoey (2016) vergleichen Eurovision-Song-Contest-Texte mit breiter stratifizierten Korpora. Diskurse von Weiblichkeit und Männlichkeit in Popsongs untersucht Kreyer (2015); Nishina (2017) setzt sprachexterne Faktoren wie Musikgenre und Geschlecht der Interpreten in Bezug zu linguistisch motivierten Analysen (Type Token Ratio, n-Gramme usw.) und kompiliert ein privates Untersuchungskorpus aus Billboard-Songs einer Dekade. Eiter (2017) untersucht Songtexte als Phänomen zwischen gesprochener und geschriebener Sprache. Ergänzend zu solchen übergreifenden Beiträgen finden sich stilistische Analysen einzelner Autoren, etwa von Johnson und Larson (2003) zur Verwendung von Metaphern in Beatles-Texten oder von Morini (2013) zu sprachlichen Eigenheiten in den Songtexten von Kate Bush.

Nicht selten werden Popsongs und ihre Texte als Spiegel gesellschaftlicher Entwicklungen betrachtet (Shukers 1998). Anderson et al. (2003) beschäftigen sich mit Korrelationen aggressiver Handlungen und der Konsumation von als aggressiv klassifizierten Texten. Machin (2010) analysiert Songtexte vor dem Hintergrund aktueller

Diskussionen um Sexualität und geschlechtergerechte Sprache. Eine diachrone Perspektive nehmen Napier/Shamir (2018) ein und beziffern mithilfe quantitativer Maße emotionale Veränderungen in Songtexten der zurückliegenden Dekaden seit 1950. Ihre Ergebnisse weisen einen langfristig signifikanten Anstieg der Kategorien Ärger, Wut und Trauer (mit einem kurzzeitigen Rückgang Mitte der 1980er-Jahre) nach. Der Ausdruck von Angst nimmt bis in die 1980er-Jahre hinein ebenfalls kontinuierlich zu, allerdings mit geringerer Steigerungsrate. Deutlich zurückgegangen über den Gesamtzeitraum ist der Ausdruck von Freude.

In jüngerer Zeit kommen verstärkt computerlinguistische Methoden und Werkzeuge für Text Mining, Sentiment Analysis oder Topic Modeling zum Einsatz. Mahedero et al. (2005) evaluieren die Eignung von Natural Language Processing-Tools für die Auswertung von Popmusiktexten; Liske (2018) beschreibt den Einsatz der Statistikumgebung R für die Analyse von Songtexten des Künstlers Prince. Penaranda (2006) verwendet Text Mining für empirisch begründete Genre-Zuordnungen auf Basis sprachlicher Auffälligkeiten.

3 Korpusdesign und -aufbereitung

Eine Grundvoraussetzung solider empirischer Erforschung sprachimmanenter Phänomenbereiche ist die technisch-physische Integrität der Primärdaten. Insbesondere der Nachweis statistischer Regularitäten hat unter Beachtung strikter Gültigkeitsbedingungen zu erfolgen, zu denen die Gewährleistung intakter Forschungsobjekte zählt (Schneider 2019, 32f.). So lassen sich auf Häufigkeitsverteilungen, Längenmessungen etc. basierende Gesetzmäßigkeiten der Textebene nachweislich nicht unter Zuhilfenahme von willkürlich kompilierten Fragmentsammlungen aus Verszeilen oder Sätzen nachweisen. Zu diesen quantitativen Korrelationen zählen Verteilungsgesetze wie das Zipf-Mandelbrot-Gesetz über den Zusammenhang zwischen Häufigkeitsrang und Frequenz lexikalischer Einheiten, funktionale Gesetze wie das Menzerathsche Gesetz über den Zusammenhang zwischen der Länge eines sprachlichen Konstrukts und der Länge seiner unmittelbaren Komponenten, oder Entwicklungsgesetze wie das Piotrovskiy-Altmann-Gesetz zur Bestimmung der Verwendungshäufigkeiten sprachlicher Einheiten aus diachroner Perspektive (vgl. Köhler 2005, Biemann 2007). Die Erklärungskraft all dieser Korrelationen entfaltet sich erst bei der Analyse zusammenhängender und ungekürzter Texte, da die

Messgrößen (Wort-, Morphem- oder Phoneminventar, Strophen- und Verszeilenlängen usw.) stets das Resultat individueller Textgenerierungsprozesse sind (Sinclair 2005).

Ziel des Korpusaufbaus ist deshalb die möglichst umfassende Abdeckung kompletter Werke. Intern fächert sich das Songkorpus auf in autoren-spezifische Archive wie das initiale Udo-Lindenberg-Archiv und themenspezifische Archive, beispielsweise eine als Chart-Song-Archiv firmierende Sammlung sämtlicher deutschsprachigen Top-100-Songtexte der zurückliegenden 20 Jahre.

Besondere Aufmerksamkeit verdient die Nutzungs- und Urheberrechtsproblematik: Grundlage des Schutzes schöpferischer Leistungen in Form von Songtexten ist das Urheberrechtsgesetz (UrhG); nach § 1 UrhG erstreckt sich der Schutz auf Werke der Literatur, Wissenschaft und Kunst. Zwar bestehen seit 2018 durch das Urheberrechts-Wissensgesellschafts-Gesetz großzügigere Regelungen für Forschungs- und Bildungseinrichtungen, trotzdem bleibt für die öffentliche Bereitstellung geschützter Inhalte über Recherche-Schnittstellen eine explizite Autorisierung der Nutzungsrechte erforderlich. Im Rahmen des Songkorpus-Aufbaus werden deshalb für öffentlich zugängliche Archive entsprechende Übertragungsvereinbarungen mit den Rechteinhabern getroffenen.

Zur Gewährleistung der Interoperabilität erfolgt die Kodierung der Songtexte mittels standardisierter Strukturbeschreibungen gemäß TEI P5 (TEI Consortium 2019), die spezielle Elementtypen für Strophen und Verszeilen bereitstellen. Nach der aufwändigen Segmentierung in Token, Verszeilen, Strophen und Sätze – Songtexte müssen primär akustisch funktionieren und enthalten deshalb selten Interpunktionszeichen zur Identifizierung von Sinneinheiten wie Phrasen und Sätzen – schließt sich eine Anreicherung um Annotations für interdisziplinäre Fragestellungen an:

- Lemmata
- Wortklassen, Morphologie, Syntax
- Neologismen bzw. originelle Produkte von Wortbildungsprozessen
- Named Entities als Identifizierung von realen und fiktiven Personen, Figuren, Institutionen, Ortsnamen etc.
- Reimformen und Reimschemata

Die Adaption von an standardnaher Sprache orientierten Kategorien und Verfahren an weniger homogene Sprachvarietäten erfordert spezifische Anpassungen (Horbach et al. 2014, Karlova-Bourbonus et al. 2016, Zinsmeister et al. 2014); Songtexte machen hier keine Ausnahme. Exemplarisch seien Konstruktionen ohne Subjekt (*hab*

noch Sehnsucht) sowie kontraktierte Formen von Verb und Personalpronomen (*machste*) oder Vergleichskonjunktion und Artikel (*wie'n*) genannt; die im Songkorpus angetroffene Vielfalt übersteigt diesbezüglich noch die in Westpfahl (2014) für den Bereich der Computer Mediated Communication (CMC) diskutierte Liste.

Insgesamt findet sich in den Texten häufig ein bewusstes Spiel mit Normen auf vielfältigen linguistischen Ebenen (Satzstrukturen, Schreibung, Semantik, Wortarten, Wortbildung etc.). Aus diesem Grund erfolgt die Korpusaufbereitung als Wechselspiel zwischen automatisierten Annotationsläufen und manueller Nachbearbeitung. Zunächst wird auf eine für das Songkorpus maßgeschneiderte Toolchain der CLARIN-Infrastrukturkomponente WebLicht (Hinrichs et al. 2010) zurückgegriffen, bestehend aus IMS-Tokenizer, TreeTagger mit STTS-Tagset (Schiller 1999), einem auf TuebaDZ trainierten Named Entity Recognizer sowie dem Berkeley Constituent Parser. Für die Kontrolle und ggf. Korrektur der Resultate erfolgt deren Import in die kollaborative Korpusplattform WebAnno (Eckart de Castilho et al. 2016). Dort kommen dann, neben einem um Phänomene der konzeptionellen Mündlichkeit erweiterten Wortklassen-Tagset (basierend auf Bartz et al. 2014, Beißwenger et al. 2015, Rehbein et al. 2012, Westpfahl et al. 2017) auch Layer und Tagsets für die Auszeichnung von Named Entitäten (basierend auf Benikova et al. 2014), Neologismen (z. B. Neuwort, Neubedeutung, Wortkombination) und Reimformen (z. B. Anfangsreim, Binnenreim, Endreim) zum Einsatz. Sämtliche manuellen Bearbeitungsschritte unterliegen während des Kurationsprozesses einer finalen Bewertung unter Zuhilfenahme von Verfahren für die Inter-Annotator-Reliabilität (Kappa-Statistiken).

4 Deskriptive Statistiken und Analysen

Das Udo-Lindenberg-Archiv versammelt mehr als 300 Texte des Pioniers der deutschsprachigen Rock- und Popsongs – und damit sämtliche nicht-fremdsprachigen Texte des Autors aus fünf Jahrzehnten sowie einzelne unveröffentlichte Songs.

	Lindenberg-Archiv	Chart-Song-Archiv
<i>Songtexte</i>	301	684
<i>Wortformen</i>	62.807	244.276
<i>Verszeilen</i>	10.688	37.734
<i>Strophen</i>	1.769	5.803

Tabelle 1. Archive im Songkorpus (Stand 10/2019).

In den zurückliegenden Jahren wurden für die Komplexität literarischer Texte verschiedene Maße und Methoden vorgestellt; vgl. z. B. Gries (2016), Perkuhn et al. (2012). Ein besonders für angewandte Disziplinen wie die Stilometrie interessanter Untersuchungsbereich betrifft Messungen zum Reichtum des Vokabulars (Yule 1944) bzw. der lexikalischen Vielfalt (Carroll 1938). Die Idee der Wortschatzvarianz geht dabei von der Annahme aus, dass gemessene Werte (Type-Token-Verhältnis als Quotient aus Type-Anzahl und Token-Anzahl) Indikatoren für den Wortschatzumfang eines Autors und mithin charakteristische Eigenschaften sind (Tanaka-Ishii/Aihara 2015). Ein methodisches Problem bleibt der Umstand, dass beinahe alle Ansätze (wie z. B. TTR, STTR) als Konsequenz des Zipf-Mandelbrot-Gesetzes (Mandelbrot 1953) abhängig von der Korpusgröße variieren (Tweedie/Baayen 1998, Evert et al. 2017). Die Online-Plattform des Songkorpus¹ bietet hierzu neben den Primärdaten verschiedene Maße und visualisierte Statistiken an.



Bild 1. Neologismen im Udo-Lindenberg-Archiv.

Zu den weiteren unmittelbar abfragbaren Daten zählen Frequenzlisten (interessanterweise finden sich hier die Wörter „und“ und „ich“ auf den vordersten Rängen, dann erst gefolgt von Artikeln), Neologismen (vgl. Bild 1), die Überprüfung quantitativer Regularitäten wie dem Zipf'schen Gesetz oder der Korrelation zwischen Strophen- und Verszeilenzahl (vgl. Bild 2) sowie Kollokationsanalysen und n-Gramme (vgl. Bild 3). Außerdem werden Ortsbezeichnungen (Named Entities) aus den Texten auf einer geografischen Karte verortet.

Bild 4 kontrastiert Worthäufigkeiten im Lindenberg-Archiv und in einem regional und zeitlich ausgewogenen allgemeinsprachlichen Korpus (zu dessen Stratifizierung vgl. Bubenhofer et al. 2013). Dabei gruppieren sich Wörter mit ähnlichen Frequenzen in beiden Sammlungen („akzeptieren“, „besonders“, „in“) nahe der zentralen

Trennlinie, während spezifische Wörter (im Lindenberg-Archiv etwa „abgefickt“, „Freund“, „Welt“) einen größeren Abstand aufweisen.

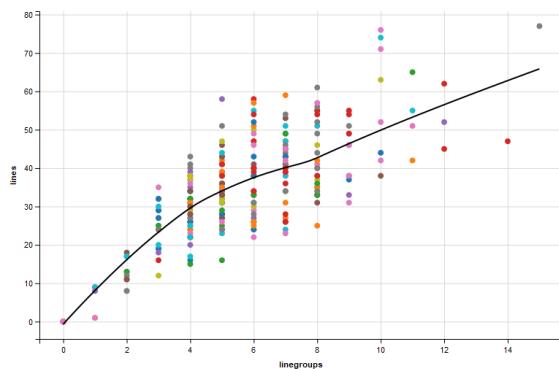


Bild 2. Strophen und Verszeilen ausgewählter Alben.

5 Fazit und Ausblick

Songtexte können als Textgattung betrachtet werden, die als eine Art "Vermündlichung des Lyrischen" Merkmale sowohl des geschriebenen als auch des gesprochenen Diskurses aufweist, sowie als Datenquelle im Kontinuum zwischen Standard und Nonstandard. Vielversprechend erscheinen gezielte Analysen sprachlicher Phänomene, die sich von Entsprechungen in anderen literarischen Schriften, Sach- und Gebrauchstexten oder spontan gesprochener Alltagssprache unterscheiden.

Das Songkorpus komplementiert den Kanon korpuslinguistischer Sammlungen um mehrfach annotierte deutschsprachige Songtexte, mit dem vorgestellten Udo-Lindenberg-Archiv sowie einem Chart-Song-Archiv als initialen Inhalten. Beide werden kontinuierlich aktualisiert und um weitere Archive ergänzt. Die TEI-annotierten Inhalte des Lindenberg-Archivs sind über das Online-Frontend recherchier- und einsehbar und lassen sich für die weiterführende wissenschaftliche Forschung gesammelt herunterladen. Ausgewählte korpuslinguistisch motivierte Auswertungen und Visualisierungen beider Archive können auf Zeichen-, Wort- und Versebene unmittelbar unter <http://songkorpus.de> berechnet werden.

Forschungsthemen, die durch die neue Ressource befördert werden, umfassen z.B.: (a) Topic Modeling, Identifizierung prominenter Themen für ausgewählte Zeiträume und Autoren (b) Parallelitäten zwischen Personen-, Orts- oder Institutionsbezeichnern und prominenten Themen im öffentlichen Diskurs (c) Sentiment Analysis zur Beschreibung von Emotionalität in Songtexten oder

¹ <http://songkorpus.de> unter dem Menüpunkt „Explorer“

Musikgenres (d) Einfluss sprachexterner Faktoren (z. B. individuelle Veröffentlichungsproduktivität) auf die lexikalische Vielfalt (e) Stilistische Analysen, Identifizierung von „style markers“ wie Verwendungshäufigkeit bestimmter Personalpronomen (f) Textähnlichkeitsmessungen (g) Reimformen und Reimschemata (h) Identifizie-

rung autoren-/zeitspezifischer Formulierungsmuster und symbolischer Elemente/Metaphern (i) empirische Annäherungen an Phänomene wie Ironie und Wortwitz (j) Variationsstudien zu dialektalen Songtexten (k) Empirische Aussagen zur Standardkonformität und Verortung im Kontinuum zwischen Schrift- und Umgangssprache.



Bild 3. Prominente Bigramme ausgewählter Alben.

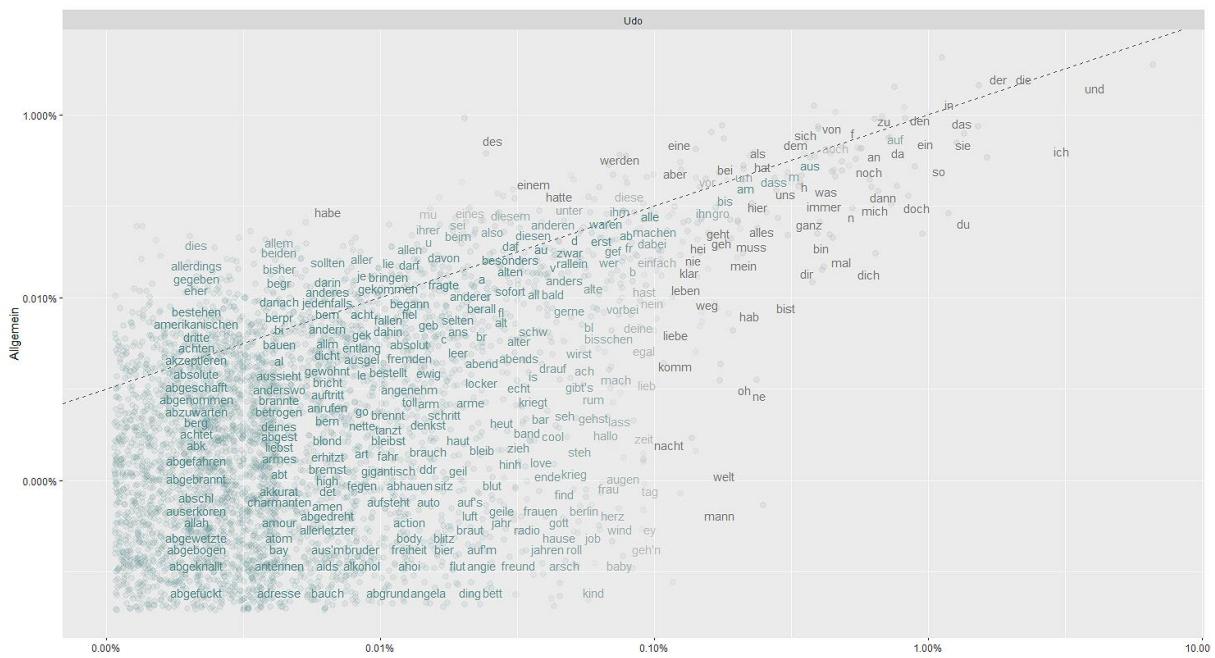


Bild 4. Wortfrequenzvergleich.

Das Songtextkorpus schließt damit eine Datenlücke, die bislang die empirisch fundierte Beantwortung syntaktischer, semantischer oder pragmatischer Fragestellungen für diese Textsorte erschwert. Die interdisziplinären Anknüpfungs-

punkte erscheinen vielfältig und vielversprechend: Neben Linguistik und Literaturwissenschaft lassen sich profitierende Forschungsbereiche im breiten Spektrum der Kulturwissenschaften sowie der Musik-, Medien- oder Geschichtswissenschaft verorten.

Literatur

- Frieder von Ammon, Dirk von Petersdorff (Hg.). 2019. *Lyrik/ lyrics. Songtexte als Gegenstand der Literaturwissenschaft*. Wallstein Verlag, Göttingen.
- Craig A. Anderson, Nicholas L. Carnagey, Janie Eubanks. 2003. *Exposure to violent media: The effects of songs with violent lyrics on aggressive thoughts and feelings*. In: *Journal of Personality and Social Psychology*, 84(5), 960–971.
- Annette Blühdorn. 2003. *Pop and Poetry – Pleasure and Protest: Udo Lindenberg, Konstantin Wecker and the Tradition of German Cabaret*. In: *German Linguistic and Cultural Studies*, Bd 13.
- Noah Bubenhofer, Marek Konopka, Roman Schneider. 2013. *Präliminarien einer Korpusgrammatik*. Korpuslinguistik und interdisziplinäre Perspektiven auf Sprache (CLIP) 4. Tübingen: Narr.
- Thomas Bartz, Michael Beißwenger, Angelika Storrer. 2014. *Optimierung des Stuttgart-Tübingen-Tagset für die linguistische Annotation von Korpora zur internetbasierten Kommunikation: Phänomene, Herausforderungen, Erweiterungsvorschläge*. In: *Journal for Language Technology and Computational Linguistics* 28 (1): 157–198.
- Michael Beißwenger, Thomas Bartz, Angelika Storrer, Swantje Westpfahl. 2015. *Tagset und Richtlinie für das Part-of-Speech-Tagging von Sprachdaten aus Genres internetbasierter Kommunikation*. Empirikom shared task on automatic linguistic annotation of internet-based communication (EmpiriST 2015). <http://sites.google.com/site/empirst2015/>
- Darina Benikova, Christian Biemann, Marc Reznicek. 2014. *NoSta-D Named Entity Annotation for German: Guidelines and Dataset*. In: *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2014)*, Reykjavík. http://www.lrec-conf.org/proceedings/lrec2014/pdf/276_Paper.pdf
- Thierry Bertin-Mahieux, Daniel Ellis, Brian Whitman, Paul Lamere. 2011. *The Million Song Dataset*. In: *Proceedings of the 12th International Society for Music Information Retrieval Conference*.
- Chris Biemann. 2007. *A Random Text Model for the Generation of Statistical Language Invariants*. In: *Proceedings of HLT-NAACL-07*. Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics. Rochester, NY, USA. <http://wortschatz.uni-leipzig.de/~cbiemann/pub/2007/biemannRandomText-HLTNAACL07main.pdf>
- John B. Carroll. 1938. *Diversity of Vocabulary and the Harmonic Series Law of Word-frequency Distribution*. In: *The Psychological Record*. 2, 16: 379–386.
- Brian Cullen. 2009. *A Corpus Analysis of Pop Song Lyrics. New Directions*. Nagoya Institute of Technology.
- Richard Eckart de Castilho, Éva Mújdricz-Maydt, Seid Muhie Yimam, Silvana Hartmann, Iryna Gurevych, Annette Frank, Chris Biemann. 2016. *A Web-based Tool for the Integrated Annotation of Semantic and Syntactic Structures*. In: *Proceedings of the LT4DH workshop at COLING 2016*, Osaka. <https://www.clarin-d.net/images/lt4dh/pdf/LT4DH11.pdf>
- Alexander Eiter. 2017. *'Haters gonna Hate': A Corpus Linguistic Analysis of the Use of Non-Standard English in Pop Songs*. University of Innsbruck, Department of English Studies. DOI: 10.13140/RG.2.2.31181.33763
- Stefan Evert, Sebastian Wankel, Elmar Nöth. 2017. *Reliable measures of syntactic and lexical complexity: The case of Iris Murdoch*. In: *Proceedings of the Corpus Linguistics 2017 Conference*, Birmingham, UK. <http://purl.org/stefan.evert/PUB/EvertWankelNoeth2017.pdf>
- Johanna Falk. 2013. *We Will Rock You: A Diachronic Corpus-based Analysis of Linguistic Features in Rock Lyrics*. Växjö: Linnaeus University.
- Reinhard Flender, Hermann Rauhe. 1989. *Popmusik: Aspekte ihrer Geschichte, Funktionen, Wirkung und Ästhetik*. Darmstadt: Wissenschaftliche Buchgesellschaft.
- Stefan Th. Gries. 2016. *Quantitative Corpus Linguistics with R*. 2nd rev. & ext. Edition. London & New York: Routledge, Taylor & Francis Group.

- Marie Hinrichs, Thomas Zastrow, Erhard Hinrichs. 2010. *WebLicht: Web-based LRT Services in a Distributed eScience Infrastructure*. In: Proceedings of the Seventh conference on International Language Resources and Evaluation. In: Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2010), Malta. http://www.lrec-conf.org/proceedings/lrec2010/pdf/270_Paper.pdf
- Andrea Horbach, Diana Steffen, Stefan Thater, Manfred Pinkal. 2014. *Improving the performance of standard part-of-speech taggers for computer-mediated communication*. In: Proceedings of KONVENTS 2014, Hildesheim, Germany.
- Mark L. Johnson, Steve Larson. 2003. ‘*Something in the Way She Moves*’: Metaphors of musical motion. In: *Metaphor and Symbol* 18(2): 63–84
- Natalie Karlova-Bourbonus, Holger Grumt Suárez, Henning Lobin. 2016. *Compilation and Annotation of the Discourse-structured Blog Corpus for German*. In: Proceedings of the 4th Conference on CMC and Social Media Corpora for the Humanities, Ljubljana.
- Noah Katzenelson, Joseph Gelman, Katrin Lindblom, Marie Caput. 2010. *American Song Lyrics: A Corpus-Based Research Project Featuring Twenty Years in Rock, Pop, Country and Hip-Hop*. San Francisco, CA: San Francisco State University.
- Reinhard Köhler. 2005. *Korpuslinguistik. Zu wissenschaftstheoretischen Grundlagen und methodologischen Perspektiven*. In: LDV-Forum, Band 20/2: 1–16. https://jlcl.org/content/2-al-lissues/22-Heft2-2005/Reinhard_Koehler.pdf
- Rolf Kreyer. 2015. “*Funky fresh dressed to impress*”: A corpus-linguistic view on gender roles in pop songs. In: *International Journal of Corpus Linguistics*, 20 (2): 174–204.
- Rolf Kreyer, Joybrato Mukherjee. 2007. *The Style of Pop Song Lyrics: A Corpus-linguistic Pilot Study*. In: *Anglia - Zeitschrift für englische Philologie*, Band 125, Heft 1: 31–58. DOI: 10.1515/ANGL.2007.31
- Mark Kupietz, Thomas Schmidt. 2018. *Korpuslinguistik*. Germanistische Sprachwissenschaft um 2020. Band 5. Berlin: Walter de Gruyter.
- Lothar Lemnitzer, Heike Zinsmeister. 2015. *Korpuslinguistik. Eine Einführung*. Tübingen: Narr.
- Debbie Liske. 2018. *Lyric Analysis with NLP & Machine Learning with R*. DataCamp. <https://www.datacamp.com/community/tutorials/R-nlp-machine-learning>
- Anke Lüdeling, Merja Kytö (Hgg.). 2008. *Corpus Linguistics. An International Handbook*. Handbücher zur Sprach- und Kommunikationswissenschaft 29 (1-2). Berlin: de Gruyter.
- David Machin. 2010. *Analysing Popular Music: Image, Sound, Text*. Los Angeles, CA: Sage.
- Jose Mahedero, Álvaro Martínez, Pedro Cano, Markus Koppenberger, Fabien Gouyon. 2005. *Natural language processing of lyrics*. In: Proceedings of the 13th annual ACM international conference on Multimedia (MULTIMEDIA '05). ACM, New York, NY: 475–478. DOI: <https://doi.org/10.1145/1101149.1101255>
- Benoît Mandelbrot. 1953. *An information theory of the statistical structure of language*. In: W. Jackson (Hg.): *Communication Theory*. New York: Academic Press: 503–512.
- Ulrich Miethaner. 2005. *I can look through muddy water: Analyzing Earlier African American English in Blues Lyrics (BLUR)*. Regensburger Arbeiten zur Anglistik und Amerikanistik 47. Frankfurt am Main: Peter Lang.
- Massimiliano Morini. 2013. *Towards a musical stylistics: movement in Kate Bush’s “Running up that Hill”*. In: *Language and Literature* 22 (4): 283–97.
- Heiko Motschenbacher. 2016. *A corpus linguistic study of the situatedness of English pop song lyrics*. In: *Corpora* 11.1: 1–28
- Tim Murphrey. 1992. *The Discourse of Pop Songs*. In: *TESOL Quarterly* 26: 770–774.
- Kathleen Napier, Lior Shamir. 2018. *Quantitative Sentiment Analysis of Lyrics in Popular Music*. In: *Journal of Popular Music Studies*, Vol. 30 No. 4, December 2018: 161–176. DOI: 10.1525/jpms.2018.300411
- Yasunori Nishina. 2017. *A Study of Pop Songs based on the Billboard Corpus*. In: *International Journal of Language and Linguistics* 4 (2) 2017: 125–134.
- Jerome Penaranda. 2006. *Text Mining von Songtexten*. Diplomarbeit. Technische Universität Wien.
- Rainer Perkuhn, Holger Keibel, Marc Kupietz. 2012. *Korpuslinguistik*. Paderborn: Fink.

- Axel Plitsch. 1997. *Music + Song = Authentic Listening in the Language Classroom*. In: Der Fremdsprachliche Unterricht Englisch 31 (1): 4–13.
- Ines Rehbein, Sören Schalowski, Heike Wiese. 2012. *Erweiterung des STTS für gesprochene Sprache*. STTS Workshop am IMS Stuttgart.
- Anne Schiller, Simone Teufel, Christine Stöckert. 1999. *Guidelines für das Tagging deutscher Textcorpora mit STTS (Kleines und großes Tagset)*. University of Stuttgart: Institut für Maschinelle Sprachverarbeitung (IMS).
- Roman Schneider. 2019. *Mehrzahl annotierte Textkorpora. Strukturierte Speicherung und Abfrage*. Korpuslinguistik und interdisziplinäre Perspektiven auf Sprache (CLIP) 8. Tübingen: Narr.
- Roy Shuker. 1998. *Key Concepts in Popular Music*. London: Routledge.
- John Sinclair. 2005. *Corpus and Text: Basic Principles*. In: Martin Wynne (Hg.): *Developing Linguistic Corpora: A Guide to Good Practice*. Oxford: Oxbow Books: 1–16.
- Kumiko Tanaka-Ishii, Shunsuke Aihara. 2015. *Computational Constancy Measures of Text. Yule's K and Rényi's Entropy*. In: Computational Linguistics 41 (3): 481–502.
- TEI Consortium (Hg.). 2019. TEI P5: *Guidelines for Electronic Text Encoding and Interchange 3.5.0*. <http://www.tei-c.org/Guidelines/P5/>
- Todd Terhune. 1997. *Pop Songs: Myths and Realities*. In: The English Connection 1 (1): 8–12.
- Fiona J. Tweedie, Harald Baayen. 1998. *How variable may a constant be?* In: Computers and the Humanities 32: 323–352.
- Thomas Van Hoey. 2016. *'Love love peace peace': a corpus study of the Eurovision Song Contest*. Graduate Institute of Linguistics, National Taiwan University.
- Claus-Ulrich Viol. 2000. *A Crack in the Union Jack? National Identity in British Popular Music*. In: Diller, H.; Otto, E.; Stratmann, G. (Hg.) (2000): *Youth Identities: Teens and Twens in British Culture*. Heidelberg: Winter: 81–106
- Ayano Watanabe. 2018. *A Style of Song Lyrics: The Case of Really*. In: Zephyr (2018), 30: 12–27. <https://doi.org/10.14989/233019>
- Valentin Werner. 2012. *Love is all around: a corpus-based study of pop lyrics*. In: *Corpora* 7 (1), S. 19–50.
- Valentin Werner, Maria Lehl. 2015. *Pop lyrics and language pedagogy: A corpus-linguistic approach*. In: Formato, F.; Hardie, A. (Hg.) (2015): *Corpus Linguistics*. Lancaster: UCREL: 341–343.
- Swantje Westpfahl. 2014. *STTS 2.0? Improving the Tagset for the Part-of-Speech-Tagging of German Spoken Data*. In: Proceedings of LAW VIII – The 8th Linguistic Annotation Workshop. Association for Computational Linguistics (ACL Anthology W14-49): 1–10. <http://www.aclweb.org/anthology/W14-4901>
- Swantje Westpfahl, Thomas Schmidt, Jasmin Jönitz, Anton Borlinghaus. 2017. *STTS 2.0. Guidelines für die Annotation von POS-Tags für Transkripte gesprochener Sprache in Anlehnung an das Stuttgart Tübingen Tagset (STTS)*. Arbeitspapier. Mannheim: Institut für Deutsche Sprache. urn:nbn:de:bsz:mh39-60634
- George Udny Yule. 1944. *The Statistical Study of Literary Vocabulary*. Cambridge University Press, Cambridge.
- Heike Zinsmeister, Ulrich Heid, Kathrin Beck. 2014. *Adapting a part-of-speech tagset to non-standard text: The case of STTS*. In: Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2014), Reykjavik. http://www.lrec-conf.org/proceedings/lrec2014/pdf/721_Paper.pdf

Combining embedding methods for a word intrusion task

Finn Årup Nielsen

Cognitive Systems, DTU Compute
Technical University of Denmark
Kongens Lyngby, Denmark

Lars Kai Hansen

Cognitive Systems, DTU Compute
Technical University of Denmark
Kongens Lyngby, Denmark

Abstract

We report a new baseline for a Danish word intrusion task by combining pre-trained off-the-shelf word, subword and knowledge graph embedding models. We test fastText, Byte-Pair Encoding, BERT and the knowledge graph embedding in Wembedder, finding fastText as the individual model with the superior performance, while a simple combination of the fastText with other models can slightly improve the accuracy of finding the odd-one-out words in the word intrusion task.

In the word intrusion task, see, e.g., (Chang et al., 2009), a cognitive agent is presented with a set of words and is to determine the odd-one-out. Such a test has been used to evaluate unsupervised topic models (Chang et al., 2009) and human subjects in experimental psychology, see, e.g., (Crutch et al., 2008). The test somewhat resembles *Test of English as a Foreign Language* (TOELF), where the task is to select the semantically most similar one among four words given a query word (Turney, 2006). A convenient method (`doesnt_match`) is implemented in the distributed semantics models of Gensim (Řehůřek and Sojka, 2010; Wohlgemant et al., 2019), giving users of this Python package a straightforward way to test trained machine learning models in odd-one-out tasks.

(Nielsen and Hansen, 2017) constructed a word intrusion dataset with Danish words and evaluated how well different machine-based methods could identify the intruded word. Explicit semantic analysis and a Word2vec-based word embedding with large corpora performed the best with performances of 73% and 71%, respectively, against a random choice baseline of 25%. Since (Nielsen and Hansen, 2017), new embedding methods have appeared with pre-trained models for non-English languages, e.g., fastText (FT) (Bojanowski et al.,

2016; Grave et al., 2018), Byte-Pair Encoding (BPE) (Heinzerling and Strube, 2018), BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018), and Wembedder (W), a knowledge graph embedding based on the multilingual Wikidata knowledge base (Nielsen, 2017). We note that some of the best performing semantic models have combined corpus-based and explicit lexicon-/knowledge graph-based methods (Turney, 2006; Speer and Lowry-Duda, 2017), and we will also pursue such a combination here.

Below we will describe the Danish word intrusion task dataset used for evaluation, the applied new off-the-shelf methods, their results in terms of accuracy of detecting the odd-one-out and finally we discuss what further approaches are needed to handle the remaining misclassified cases.

1 Evaluation dataset

The word intrusion dataset comprises 100 sets of 4 words each where one of 4 is the outlier to be detected (Nielsen and Hansen, 2017),¹ see the left part of Figure 1 for a small excerpt of the dataset. The dataset contains common and proper nouns (named entities) and other word classes as well as a few numbers, years and phrases. Some sets of “words” require detailed Danish world knowledge, e.g., 1807, 1864, 1940, 1909, — the last being the outlier as the three first years relates to Danish military defeats. The dataset contains also several homographs/polysemous words. Most of the words are common nouns. There are 11 word sets with proper nouns and 11 with verbs. Further sets includes sets of adjective and other word classes. A few of the word sets mix lexical categories, e.g., the set (halvsyg, forkølelse, hoster, vej) corresponding to the English (“half-sick”, flu, the verb “coughs”, road).

¹https://github.com/fnielsen/dasem/blob/master/dasem/data/four_words_2.csv. We use the second version correcting two spelling errors.

word 1	Word 2	word 3	word 4	FT	BERT	W	FT+W+BERT
æble (apple)	pære (pear)	kirsebær (cherry)	stol (chair)	stol	kirsebær	kirsebær	stol
bil (car)	cykel (bike)	tog (train)	vind (wind)	tog	bil	bil	tog
Finland (Finland)	Sverige (Sweden)	Norge (Norway)	Kina (China)	Kina	Norge	Kina	Kina
tres (sixty)	60	LX (LX)	3 (3)	tres	LX	LX	LX

Table 1: Excerpt of the evaluation dataset and individual results from fastText (FT), BERT, Wembedder (W) and the combined system of fastText, BERT and Wembedder (FT+W+BERT). The ground truth outlier is in the *word 4* column.

While word intrusion tasks may be based on the sound of words, see, e.g., (Oakhill et al., 2003), the Danish dataset contains none of this kind, so the methods we employ need no phonological information.

2 Methods

We use fastText (Bojanowski et al., 2016; Grave et al., 2018) through the Gensim 3.6.0 implementation (Řehůřek and Sojka, 2010) with the fastText cc.da.300.bin pre-trained model.² This model has been trained on the *Common Crawl* and the *Danish Wikipedia* with the continuous bag-of-words setup. In terms of training corpus size, it may be the largest publicly available linear word embedding model and as such should be regarded as a baseline model. We downloaded it from its homepage.³

For BERT, we use the currently recommended cased multilingual model⁴ through the package bert-as-service.⁵

The BPE model comes in various sizes of vocabulary and embedding dimensions and we test them all.⁶ The size of the vocabulary of the pre-trained distributed models ranges from 1,000 to 200,000 while the embedding dimension ranges from 25 to 300.

²<https://fasttext.cc/docs/en/crawl-vectors.html>.

³<https://fasttext.cc/docs/en/crawl-vectors.html>

⁴ multi_cased_L-12_H-768_A-12 from <https://github.com/google-research/bert/blob/master/multilingual.md>

⁵<https://github.com/hanxiao/bert-as-service>

⁶<https://nlp.h-its.org/bpemb/da/>.

Wembedder is an embedding of Wikidata items rather than words, and the use of Wembedder for natural language requires a translation from the word to the Wikidata item identifier. We use the Wikidata search API⁷ and its wbsearchentities action to search for Wikidata items based on the queried word or phrase. Not all words can be found in Wikidata, e.g., adjectives and verbs are rarely present as Wikidata items, meaning words from such word classes are usually out-of-vocabulary. The Wembedder model we use is the one trained on the 2017-06-13 truthy dump of Wikidata with an embedding dimension of 100 and using the continuous bag-of-word Word2vec approach implemented in Gensim.⁸ The use of the Wikidata API means that results may not necessarily reproduce between runs of our evaluation, because Wikidata is continuously expanded and modified.

There are multiple ways of getting from a vectorial representation to a measure of outliers. For Gensim-based models, we use Gensim’s doesn’t_match method. For the other embeddings, we sort the row sum of the correlation matrix of the concatenated embedding vectors of the four words and select the word associated with the lowest sum. The performance of a model is measured as the percentage of correctly detected outliers.

Our computations are available in a public Jupyter Notebook.⁹

⁷<https://www.wikidata.org/w/api.php>

⁸The Wembedder Gensim model was downloaded from <https://zenodo.org/record/827339>.

⁹<https://gist.github.com/fnielsen/93f3b68941e74c468522f187e2dbe9a7>.

Model	FT	BPE	BERT	W	FT+W	FT+W+BERT	Random
Accuracy	78	64	32	47	82	83	25

Table 2: Odd-one-out detection percentage for fastText (FT), BPE, BERT, Wembedder (W), fastText and Wembedder (FT+W) and the combined model of fastText, Wembedder and BERT (FT+W+BERT) against the random choice.

3 Results

The results are displayed in Table 2. FastText alone can improve the benchmark to 78%, while the BPE embeddings cannot reach a better performance than our previous results. Its accuracies range from 33% to 69%, depending on dimension and vocabulary. Generally, the performance increases considerably as the vocabulary increases, see Table 3. However, for the largest vocabulary (200,000) the accuracy decreases for the models with the largest embedding dimension. With respect to the dimension, the largest embeddings with sizes 200 and 300 yield the best performance. The increase in performance from low to high dimensional models is smaller than when the vocabulary size is changed. This difference could be explained by the different range: The vocabulary sizes differs by 200 times, while the embedding dimension only differ by 12 times.

Our current simple application of BERT does not yield good performance with only an accuracy of 32%.

Wembedder neither performs well with just 47% accuracy. However, it tends to perform well on proper nouns, better than (sub-)word embedding models: We can attain an accuracy of 82% by combining fastText and Wembedder using Wembedder for entries with non-lower first letters (named entities). We can improve that performance slightly to 83% by using BERT for phrases which are not named entities (as we only have 100 tests these improvements are not statistically strong).

4 Discussion

Our best model detects 83 outliers out of 100. What is needed to improve the performance, handling the misclassified cases?

The 17 errors made form a heterogeneous set. A handful of them may well be due to homographs, e.g., ‘tog’ (either ‘train’ or ‘took’) and ‘kassen’ (‘the box’), where the Wembedder search identifies the latter as the surname ‘Kassen’ ([Q37436530](#)) for the set (Nielsen, Jensen, Olsen, kassen). If we are to improve the model, it may be necessary to

Voc. \ Dim.	25	50	100	200	300
1,000	36	34	34	36	33
3,000	45	42	48	47	47
5,000	52	50	51	54	55
10,000	56	59	59	63	59
25,000	58	58	62	63	67
50,000	58	63	65	69	69
100,000	58	63	63	69	69
200,000	60	64	67	67	64

Table 3: BPE results. Percentage of correctly spotted outliers among four words for BPE models of varying sizes: vocabulary from 1,000 to 200,000 words and dimensions from 25 to 300.

handle the homography/polysemy of words. It is likely that even larger corpora with the non-context embedding models such as the ordinary application of fastText may not be able to handle the cases with homographs.

Numbers pose a common problem for all the models. One of the tests is (*tres*, 60, LX, 3), where *tres* is the Danish word for sixty, LX is the Latin number 60 and 3 is the outlier. FastText chooses *tres*, while BERT, the largest BPE model and Wembedder report LX as the outlier, so modifying any ensemble weighting will not help. It is possible that a larger corpora could learn the relations, or that explicit entry of such information in the Wikidata knowledge graph could help.

The low performance of BERT may come as a surprise given that BERT has been reported with a string of state-of-the-art results ([Devlin et al., 2018](#)). We note that the benchmarks used in the original BERT report had input that was longer than a word (e.g., sentences), while our current application of BERT only submits one word at a time to the model. It is tempting to think that some form of multiple word input to BERT may perform better, e.g., where two or three of the four words in a word set are submitted at a time. Such an approach could also handle the homography/polysemy problem.

The measure of outlierness is based on

the cosine similarity implemented in Gensim’s `doesnt_match` function and the correlation matrix. We note that an exploration and a more careful selection of the metric for comparison may yield different results.

Over 1,800 entities for Danish words, affixes and phrases exist as lexemes on Wikidata (Nielsen, 2019), but the current Wembedder models have no Wikidata lexemes. Knowledge graph embedding that includes the relatively new Wikidata lexemes and its connection to the Danish wordnet DanNet (Pedersen et al., 2009) may be a fruitful avenue for further study.

5 Acknowledgment

This research is funded by the Innovation Foundation Denmark through the DABAI and the ATEL projects.

References

- [Bojanowski et al.2016] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomáš Mikolov. 2016. Enriching Word Vectors with Subword Information. July. <https://arxiv.org/pdf/1607.04606.pdf>.
- [Chang et al.2009] Jonathan Chang, Jordan Boyd-Graber, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David M. Blei. 2009. Reading Tea Leaves: How Humans Interpret Topic Models. *Advances in Neural Information Processing Systems* 22, pages 288–296.
- [Crutch et al.2008] Sebastian J Crutch, Sarah Connell, and Elizabeth K Warrington. 2008. The different representational frameworks underpinning abstract and concrete knowledge: evidence from odd-one-out judgements. *Quarterly Journal of Experimental Psychology*, 62:1377–88, 1388–90, December.
- [Devlin et al.2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. October. <https://arxiv.org/pdf/1810.04805.pdf>.
- [Grave et al.2018] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomáš Mikolov. 2018. Learning Word Vectors for 157 Languages. *Proceedings of the 11th edition of the Language Resources and Evaluation Conference*, February.
- [Heinzerling and Strube2018] Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. *Proceedings of the 11th edition of the Language Resources and Evaluation Conference*, pages 2989–2993, May.
- [Nielsen and Hansen2017] Finn Årup Nielsen and Lars Kai Hansen. 2017. Open semantic analysis: The case of word level semantics in Danish. *Human Language Technologies as a Challenge for Computer Science and Linguistics*, pages 415–419, October.
- [Nielsen2017] Finn Årup Nielsen. 2017. Wembedder: Wikidata entity embedding web service. October. <https://arxiv.org/pdf/1710.04099>.
- [Nielsen2019] Finn Årup Nielsen. 2019. Danish in Wikidata lexemes. *Global WordNet Conference 2019*.
- [Oakhill et al.2003] J.V. Oakhill, K. Cain, and P.E. Bryant. 2003. The dissociation of word reading and text comprehension: Evidence from component skills. *Language and Cognitive Processes*, 18:443–468, August.
- [Pedersen et al.2009] Bolette Sandford Pedersen, Sanni Nimb, Jørg Asmussen, Nicolai Hartvig Sørensen, Lars Trap-Jensen, and Henrik Lorentzen. 2009. DanNet: the challenge of compiling a wordnet for Danish by reusing a monolingual dictionary. *Language Resources and Evaluation*, 43:269–299, August.
- [Speer and Lowry-Duda2017] Robert Speer and Joanna Lowry-Duda. 2017. ConceptNet at SemEval-2017 Task 2: Extending Word Embeddings with Multilingual Relational Knowledge. April. <https://arxiv.org/pdf/1704.03560.pdf>.
- [Turney2006] Peter D. Turney. 2006. Similarity of Semantic Relations. *Computational Linguistics*, 32:379–416, September.
- [Wohlgemant et al.2019] Gerhard Wohlgemant, Ekaterina Chernyak, Dmitry Ilvovsky, Ariadna Barinova, and Dmitry Mouromtsev. 2019. Relation Extraction Datasets in the Digital Humanities Domain and their Evaluation with Word Embeddings. March. <https://arxiv.org/pdf/1903.01284.pdf>.
- [Řehůřek and Sojka2010] Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. *New Challenges For NLP Frameworks Programme*, pages 45–50, May. https://radimrehurek.com/gensim/lrec2010_final.pdf.

Deep learning for Free Indirect Representation

Annelen Brunner, Ngoc Duyen Tanja Tu
Leibniz-Institut für Deutsche Sprache
R5 6-13
D-68161 Mannheim
brunner|tu
@ids-mannheim.de

Lukas Weimer, Fotis Jannidis
Universität Würzburg
Am Hubland
D-97074 Würzburg
lukas.weimer|fotis.jannidis
@uni-wuerzburg.de

Abstract

In this paper, we present our work-in-progress to automatically identify free indirect representation (FI), a type of thought representation used in literary texts. With a deep learning approach using contextual string embeddings, we achieve f1 scores between 0.45 and 0.5 (sentence-based evaluation for the FI category) on two very different German corpora, a clear improvement on earlier attempts for this task. We show how consistently marked direct speech can help in this task. In our evaluation, we also consider human inter-annotator scores and thus address measures of certainty for this difficult phenomenon.

1 Introduction

In contrast to the well-known direct or indirect representation of speech, thought and writing, there have been hardly any attempts to tackle the automatic recognition of free indirect representation (FI) up until now. FI – in German also known as “Erlebte Rede” – is mainly used in literary texts to represent a character’s thoughts while still maintaining characteristics of the narrator’s voice. In the following example, the part in italics is FI:

Er glaubte, sie zu kennen. *War das nicht die Grünkramfrizten von der Ecke?* [He thought he knew her. *Wasn’t that the greengrocer gal from the corner?*]

While the third person pronouns and the past tense indicate the narrator’s voice, the use of a question and the informal language makes the presentation similar to a direct quotation of the character’s thoughts. FI has been a much discussed topic in literary theory since the early 20th century (overview in McHale (2014)). In our approach we follow the ‘classical’ definition of FI (e.g. Genette

(2010), Leech and Short (2013)) that focusses on the representation of characters’ thought processes. When a personal, character-focussed style gradually became mainstream over the last century, FI became a common narrative tool which today appears even in popular and children’s literature. For quantitative studies of literary style, it would be highly useful to be able to detect the usage of FI automatically. However, FI is very context dependent, essentially a shift in perspective to the character, which can be hard to detect even for humans. In this brief presentation of our work-in-progress in this area, we will show preliminary results with a deep learning approach which we will contrast with a simple rule-based approach as well as a RandomForest approach from earlier research. We will also consider human inter-annotator agreement in our evaluation.

2 Related Work

The only attempt of automatic detection of FI in German texts known to us is the work by Brunner (2015). She implemented a simple rule-based algorithm and also trained a RandomForest model. On a corpus of 13 short German narratives (57,000 tokens) from the late 18th to early 20th century, she reports a sentence-based f1 score for the category FI (as opposed to non-FI) of 0.31 (rule-based) and 0.4 (RandomForest, 10-fold cross validation). We will compare our results to Brunner’s in section 5.2.

With respect to automatization, we consider the detection of FI a sequence labelling task. In this area, much progress has been made recently employing deep learning and language embeddings. We use FLAIR (Akbik et al., 2019), a PYTORCH-based framework that facilitates the use of language embeddings and model training for NLP tasks. The architecture of our deep learning model is adapted from Akbik et al. (2018). They propose ‘contextual string embeddings’, an approach which passes sen-

tences as sequences of characters into a character-level language model to form word-level embeddings. This approach achieves significant improvements for NER, especially for German, and state-of-the-art results for chunking and POS tagging and can be considered one of the leading architectures for sequence labelling tasks to date. We will detail the exact configuration of our model in section 4.2.

3 Training data

As mentioned above, FI became much more common in modern times. For this reason we decided to use mainly modern popular literature for our test and training data.

The bulk of our training data comprises dime novels as well as popular crime novels (full texts or excerpts). This data was preprocessed with a basic rule-based FI recognizer (description see section 4.1). The automatically detected instances were then presented to human annotators who could either dismiss or accept them. The annotators were also instructed to annotate any additional cases of FI in the direct vicinity of the automatically detected instances. This sped up the annotation process considerably, but of course also created a bias, as it is quite possible that valid instances of FI that were never detected by the rule-based recognizer have been missed. This material was supplemented by 150 instances¹ of FI with little to no context, manually extracted from 20th century novels. For model training, it was split into a training corpus (1,443,811 tokens with 5.46% FI, 2551 instances) and a validation corpus (181,916 tokens with 3.85% FI, 205 instances).

4 Automatic approaches

4.1 Rule-based recognizer

For the rule-based FI annotation, the text is preprocessed using OpenNLPSentenceDetector, OpenNLPTokenizer (<https://opennlp.apache.org>), MateLemmatize (Björkelund et al., 2010) and TreeTagger (Schmid, 1995) with the STTS tagset (Schiller et al., 1999). Sentences that contain direct speech (as identified by a simple approach matching quotation marks) are skipped, as it is relatively unlikely for them to contain FI and they exhibit many similarities to FI at the same time. The remaining sentences are categorized as FI if

they contain any typical FI indicators, e.g. typographical markers like ! ... ? –, temporal markers indicating the present as a reference point (*gestern [yesterday]*, *heute [today]*, *morgen [tomorrow]*, *jetzt [now]*), forms of *würde [would]*) which are commonly used to refer to the future in FI, or the STTS tags ITJ (interjection) or PTKANT (modal particles). This basic recognizer was mainly used to aid in the generation of training material, but its results will serve as a baseline for our evaluation.

4.2 Deep learning model

For language embedding, we used pre-trained models provided by the FLAIR framework, combining word embeddings with contextual string embeddings as recommended in Akbik et al. (2018) in the following combination: ‘de’ (fastText word embedding (Bojanowski et al., 2016) with 300 dimensions, trained over Wikipedia), ‘german_forward’, ‘german_backward’ (two contextual string embeddings trained with a mixed corpus of web texts, Wikipedia and subtitles).

To train our tagging model, we used FLAIR’s SequenceTagger class which implements a BiLSTM-CRF architecture on top of the language embedding (as proposed by Huang et al. (2015)). After initial tests with one bidirectional LSTM layer with hidden size 256 and one CRF layer, we decided to add a second BiLSTM layer (hidden size also 256) on top of the first to account for the complexity of our task. This led to visible improvements in both precision and recall. The latter model was used to create the results presented below.

Some consideration was given to the format in which we present the data to our model. FI has a tendency to appear in blocks of several consecutive sentences and, as explained above, constitutes a shift in narrative perspective. Because of this, it is extremely difficult to identify a single sentence as FI without its context, even for humans. On the other hand, though FI most often comprises at least one sentence, it can also be shorter, if the perspective shift occurs within a sentence. We therefore opted to model the sequence labelling task on token level, but input the data as rather large chunks of up to 100 tokens, which may span several sentences. Note that the chunks can be shorter than this maximum, as they may never cross borders between different texts or cut sentences (unless a sentence is longer than 100 tokens).

¹An instance is defined as a continuous passage of FI tokens and may span several sentences.

5 Results and discussion

5.1 Evaluation on the dime novel corpus

The test data comprises 22 excerpts from dime novels (romance and horror), each about 1,000 tokens long. These were manually annotated in full by humans. To give justice to the difficult nature of FI, we present two competing annotations: **anno1** was done by a single person who annotated all excerpts; **anno2** was done by two different people, each annotating half of the excerpts.² All annotators were trained to recognize FI according to the definition used in our project, but worked independently and did not discuss this annotation. The differences between their results are mostly due to true borderline cases rather than clear mistakes. In table 1 we present the agreement scores between the two human-made annotations followed by an evaluation of our deep learning recognizer. Note that the recognizer scores are the results of one model (as described in section 4.2), compared to four different gold standards: the two different human annotations (anno1 and anno2), the set of cases agreed upon by both anno1 and anno2 (**anno_all**; i.e. cases which can be considered fairly obvious for humans) as well as the set of cases marked by either anno1 or anno2 (**anno_any**; i.e. cases that at least some humans would see as FI). This gives us a better understanding of the performance in relation to human certainty. According to anno_all, the test corpus contains 163 (9%) FI sentences, according to anno_any there are 304 (16%) FI sentences.

In addition to that, we tested for the influence of quotation marks on the results. This is relevant, because by definition, FI has many similarities to direct representation (character specific speech patterns, questions, exclamations etc.). The presence of quotation marks makes it much easier to distinguish between the two forms. We tested our recognizer on one version of our test corpus that lacked any quotation marks and one that used a consistent pattern of quotation marks. The training data marked direct speech in most but not all cases, using a variety of patterns.

Table 1 shows the agreement scores between humans, our rule-based baseline as well as the results of our deep learning model on texts with and without quotation marks. As FI is a mostly sentence-

²As the skill levels of all annotators were similar and we are not interested in the performance of any one annotator, we believe it is valid to treat this annotation as though it was done by one person as well.

	f1	prec	rec	acc
human agreement				
anno1 vs. 2	0.7	0.73	0.67	0.93
rule-based (baseline)				
anno1	0.37	0.57	0.27	0.88
anno2	0.31	0.45	0.24	0.88
anno_all	0.35	0.42	0.3	0.9
anno_any	0.34	0.61	0.23	0.85
deep learning				
(data without quotation marks)				
anno1	0.46	0.61	0.37	0.89
anno2	0.46	0.58	0.38	0.89
anno_all	0.46	0.48	0.44	0.91
anno_any	0.46	0.71	0.34	0.87
deep learning				
(data with consistent quotation marks)				
anno1	0.45	0.78	0.32	0.9
anno2	0.48	0.79	0.35	0.91
anno_all	0.49	0.65	0.39	0.93
anno_any	0.45	0.92	0.3	0.88

Table 1: F1 score, precision, recall (for category FI) and overall accuracy on the dime novel test corpus, calculated over sentences. Results reported with varying gold standards.

based phenomenon, we calculate the scores on sentences, though the recognition happened on tokens. The (very rare) cases when FI was partially recognized were counted as correct. F1, precision and recall scores are provided for the category FI.

The agreement score between human annotators, $f1=0.7$ (fleiss kappa=0.66), can serve as an indicator on how much certainty can be expected when identifying FI in general. We can see that our deep learning model clearly outperforms the rule-based baseline, regardless of quotation marks. When quotation marks are added, you can observe a strong increase in precision and some decrease in recall.

In our error analysis we focussed on cases which have not been identified as FI by the model even though both annotations agreed on them ('clear cases') and, in contrast, cases that were identified by the model even though none of the humans considered them FI ('unlikely cases').

Table 2 lists those cases and sorts them into rough categories. In general we see that the recognizer has its weakness in recall much more than in precision. This is especially true if quotation marks are present: Their absence causes the annotator to categorize direct representation as FI. The

missing clear cases (false negatives for anno_all)		
	no quotes	quotes
no indicators	57	50
known indicators	17	10
partial passage	18	39
total	92	99

finding unlikely cases (false positives for anno_any)		
	no quotes	quotes
direct speech	38	0
known indicators	2	6
overlong passage	3	2
total	43	8

Table 2: Error analysis for the two versions of the test data: ‘no quotes’ = without quotation marks; ‘quotes’ = with consistently used quotation marks

addition of quotation marks eliminates this problem completely and hardly any ‘unlikely’ cases remain.

We sorted the error cases into the following categories:

- **no indicators:** isolated sentence with no obvious FI indicators (recognizable only by context); example:³

Sie war glücklich mit dem Resultat, so viel war deutlich. Aber bestimmt nicht halb so glücklich wie er. *Sein Instinkt hatte ihn also nicht getäuscht, sie war perfekt.* [She was happy with the result. But certainly not half as happy as he was. *His instinct had not deceived him, she was perfect.*] (Perspective shift into the head of the man in the last sentence.)

- **known indicators:** isolated sentence which contains known FI indicators; these can be specific surface markers like the ones used by the rule-based recognizer, but also softer indicators like informal speech patterns; example:

»Auch das noch!« *Nicht, dass es ihn überraschte – er hatte sie von Anfang an gewarnt.* [»Oh no, not that!« *Not that it surprised him – he had warned her from the beginning.*] (Ellipsis in the first sentence part and dash, which is a known surface indicator of FI.)

³The italicized parts of the examples are FI, according to anno_all.

- **partial passage / overlong passage:** sentence adjacent to a longer passage of FI that is either not annotated or added incorrectly; example:

Er hatte sie geküsst! Und es war noch herrlicher gewesen, als sie sich erträumt hatte. [*He had kissed her! And it had been even more glorious than she had dreamed.*] (Both sentences are FI. The recognizer detected the first but not the second.)

The cases adjacent to an FI passage were categorized separately, as one can argue that these errors are less grave: The recognizer at least identified that FI is present in this part of the text, but detected the wrong borders for the passage.

It is also heartening that the recognizer only incorrectly labeled sentences as FI that had at least some known FI indicators; example:

Der Wagen auf der Achterbahn fuhr weiter. Jetzt hatte er den höchsten Punkt der Steigung erreicht. [The car on the roller coaster went on. Now it had reached the highest point of the ascent.] (The second sentence was incorrectly labeled as FI. It contains the surface indicator *jetzt* [now].)

The biggest issue, both in numbers as well as in gravity, are the missing isolated sentences, especially the cases of context-based FI without clear indicators within the sentence itself.

5.2 Evaluation on the Brunner corpus

We also tested our deep learning model on the corpus used by Brunner (2015).⁴ Table 3 shows the evaluation and contrasts them with the results of Brunner’s RandomForest model.⁵ We also provide the scores of the rule-based recognizer which were extremely poor for this corpus due to a large number of false positives in a text with unmarked dialogue and many false negatives for FI sentences without explicit indicators.

We are happy to see that our model gives comparable scores to the ones for the dime novel corpus even though Brunner’s corpus is very different:

⁴Brunner’s corpus and all her annotations are available for download at <http://hdl.handle.net/10932/00-027B-9E8A-9300-0B01-E>

⁵The scores reported for Brunner’s RandomForest model differ slightly from the ones reported in Brunner (2015), as we used a different sentence splitting tool on her corpus for easier comparison.

	f1	prec	rec	acc
rule-based	0.04	0.06	0.03	0.94
Brunner’s RF model	0.41	0.61	0.31	0.96
our model	0.52	0.65	0.43	0.96

Table 3: Scores on Brunner’s corpus, in comparison to the scores of the rule-based recognizer and of Brunner’s RandomForest model; gold standard is Brunner’s manual annotation.

It contains historical texts (1787-1913) with only partly modernized spelling and a lot of stylistic variation, while the model was trained almost exclusively on modern popular literature and uses language embeddings generated from modern German. The percentage of FI is also much lower than in the dime novel corpus, only 4.5% (99 FI sentences), and highly skewed towards one text. Still, our model clearly outperforms Brunner’s RandomForest model, which was trained on her own corpus (in 10-fold cross-validation). It looks as though the FI characteristics learned by our model are valid for more than one genre and time period. The error analysis for the Brunner corpus showed the same tendencies as for the dime novel corpus.

6 Conclusion and outlook

We presented our deep learning model for FI and evaluated it on two very different corpora with similar results. Though the f1 scores are only in the 0.45 to 0.5 range and there are problems, especially with respect to recall, they clearly outperform a rule-base detection of FI as well as a RandomForest approach. Considering that trained human annotators only achieved an f1 score of 0.7 (fleiss’ kappa 0.66), the results are promising. We also showed that the presence of quotation marks for direct representation has a strong effect on precision.

We will continue trying to improve our model. One focus is on training data: Apart from simply adding more data, we plan to add specifically more FI cases without clear surface markers in order to fix our recall problem. We also consider removing long passages without detected FI from our current training data, as due to the semi-automated annotation process those could easily contain valid FI. The second focus is on testing other leading language embeddings for this task, such as BERT (Devlin et al., 2018).

References

- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.
- Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota, June. Association for Computational Linguistics.
- Anders Björkelund, Bernd Bohnet, Love Hafell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstration Volume*, pages 33–36, Beijing.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.
- Annelen Brunner. 2015. *Automatische Erkennung von Redewiedergabe. Ein Beitrag zur quantitativen Narratologie*. Number 47 in Narratologia. de Gruyter, Berlin u.a.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.
- Gérard Genette. 2010. *Die Erzählung*. Number 8083 in UTB. Fink, Paderborn, 3 edition.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.
- Geoffrey Leech and Mick Short. 2013. *Style in fiction. A linguistic introduction to English fictional prose*. Routledge, London u.a., 2 edition.
- Brian McHale. 2014. Speech Representation. In Peter Hühn, John Pier, Wolf Schmid, and Jörg Schönert, editors, *The living handbook of narratology*. Hamburg University Press, Hamburg.
- Anne Schiller, Simone Teufel, Christine Stöckert, and Christine Thielen. 1999. *Guidelines für das Tagging deutscher Textkorpora mit STTS (Kleines und großes Tagset)*. Institut für Maschinelle Sprachverarbeitung (Universität Stuttgart) / Seminar für Sprachwissenschaft (Universität Tübingen), August.
- Helmut Schmid. 1995. Improvements on Part-of-Speech Tagging with an Application to German. In *Proceedings of the ACL SIGDAT-Workshop*, Dublin, Ireland.

Representing document-level semantics of biomedical literature using pre-trained embedding models

Jon Stevens

AbbVie Information Research
jon.stevens@abbvie.com

Derek Chen

University of Michigan School of
Information
dinganc@umich.edu

Jacob Zimmer

University of Michigan School of
Information
zimmerja@umich.edu

Brandon Punturo

AbbVie Information Research,
University of Michigan School of
Information
bpunt@umich.edu

Mike Kim

University of Michigan School of
Information
miketkim@umich.edu

Abstract

We present two novel tasks aimed at capturing document-level semantics, i.e., high-level topical or thematic content, of biomedical scientific publications. We use these tasks to evaluate whether word and sequence embedding models pre-trained on biomedical literature can be used to derive meaningful document-level semantic representations for these publications. We evaluate approaches from two broad categories: **(1) lexical pooling**, or vectorizing documents purely based on aggregation of lexical items, which includes the NCBI’s BioWordVec model and Tf-idf-based vectorizations, both with and without word pre-filtering based on biomedical ontologies, **(2) sequence embedding**, which includes the NCBI’s BioSentVec model and BioBERT. For both of our tasks, lexical pooling outperformed sequence embedding, and the best overall method was mean pooling of BioWordVec word embeddings. We also include baselines trained on non-biomedical English to show that training on biomedical literature is warranted. The methods discussed here have potential applications for clustering, comparing, analyzing and recommending scientific literature in the biomedical domain.

1 Background

The last several years of NLP research have seen a number of breakthroughs leveraging the concept of transfer learning (Pan & Yang 2010), particularly in the form of pre-trained embedding models. Such models provide low-dimensional vectorized representations of text which are informed by large corpora but can be applied to small-data NLP tasks. Example architectures include, for static word embeddings, word2vec/skip-grams (Mikolov et al. 2013), GloVe (Pennington et al. 2014) and fasttext (Bojanowski et al. 2017), for sentence and paragraph embeddings, doc2vec (Le & Mikolov 2014) and sent2vec (Pagliardini et al. 2018), and for context-sensitive word and sequence embeddings, ULM-Fit (Howard & Ruder 2018) and BERT (Devlin et al. 2018), the latter adapting the transformer architecture of Vaswani et al. (2017).

These models have found use within the biomedical domain, particularly for processing scientific literature, where the NCBI’s PubMed and MedLine databases provide a large, freely available data source for model training. Most recently, the NCBI has released two pre-trained embedding models, both trained on millions of biomedical abstracts and clinical notes (Chen et al. 2018): (1) BioWordVec, a static word embedding model based on fasttext, which uses character-level information to enhance word embeddings, particularly those of rare words, and (2) BioSentVec, a sentence

embedding model based on sent2vec, which learns to embed words and n-grams and average them to create a single sentence embedding, optimized on the task of predicting missing words.

Lee et al. (2019) have fine-tuned the base BERT model on millions of biomedical texts, including those from MedLine. This model, dubbed BioBERT, can also serve as both a contextual word embedding model and a document embedding model, if one pools the penultimate layer of transformer outputs (Xiao 2018).

Going beyond end-to-end embedding models, we may also incorporate BioNLP’s long tradition of utilizing curated biomedical vocabularies and ontologies to extract insights from literature via text mining (see Fleuren & Alkema 2015 for an overview). In our case, we use ontologies to refine some models by giving higher weight to biomedically relevant terms in the text.

While great progress has been made in the evaluation of biomedical word embeddings (see e.g. Chiu et al. 2016; Wang et al. 2018), these evaluations have been aimed, naturally, at word- and phrase-level semantics, focusing on either word similarity or downstream tasks which do not require good representations of the overall thematic or topical content of each document. Moreover, intrinsic evaluations of embedding quality tend to rely on subjective scoring or ranking, where the assumptions about the semantic space into which the documents are embedded are unclear.

2 Overview

The aim of this paper is to compare different methods for using pre-trained models to create embedded representations of scientific publications from the MedLine database, and to evaluate their ability to capture document-level semantics in a useful way. To this end, we introduce two tasks that leverage document-level semantics: prediction of academic departments from MedLine titles/abstracts, and pairwise correlation of model-derived document similarity (measured as cosine similarity of the document embeddings) with document similarity derived from gold-standard Medical Subject Headings (so-called MeSH terms). On these tasks we compare methods derived from the BioWordVec, BioSentVec and BioBERT models, as

well as n-gram Tf-idf vectorization and two embedding models pre-trained on general English. For the BioWordVec and Tf-idf methods we also evaluate the addition of biomedical ontologies to pre-select only words with biomedical relevance.

Having high-quality embedded document representations has a host of potential applications, both in the biomedical domain and in similar domains, including efficient document clustering, similarity scoring and the construction of knowledge graphs for easier discovery of scientific literature.

3 Vectorization Methods

The goal of each method is to produce a single vectorized representation of a MedLine document (here taken to be title + abstract text) which can be used to (a) determine similarity of two documents, and (b) serve as a featurization technique for machine learning models on small-data downstream tasks. To this end, we compare a number of methods belonging to two broad categories. The first is lexical pooling – documents are vectorized in a “bag of words” manner, by pooling vectorized lexical items. The lexical pooling methods tested are:

- **BioWordVec** pooling: for each word in the document text, obtain the BioWordVec embedding, then average pool all word embeddings to obtain a single document embedding
- **BioWordVec+:** BioWordVec pooling with word pre-filtering based on noun phrase dependency parsing and biomedical vocabularies (see Fig.1): pre-process the document text by extracting likely entities and biomedically relevant terms, and then pool only these word embeddings to create the document embedding. The pre-processing steps are as follows:¹
 1. Using established biomedical ontologies such as MedDRA and ChEMBL, extract all the phrases from the document which refer to concepts in these ontologies as well as the preferred names for any such concepts
 2. Using a pre-trained English dependency parser, identify and extract all of the noun phrases in the document

¹ Scibite’s TERMite platform was used to apply the ontologies, and the spaCy dependency parser was used to extract the noun phrases.

3. Concatenate the ontology phrases and concept names with the extracted noun phrases to create the input for vectorization

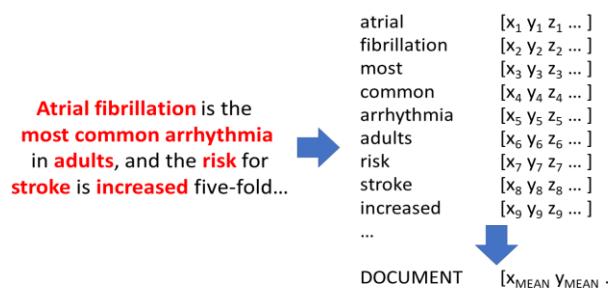


Figure 1: Illustration of BioWordVec pooling with entity extraction, one of the document vectorization methods we assess.

- High-dimensional **Tf-idf** vectorization of word and bigram tokens²
- **Tf-idf+:** Tf-idf with word pre-filtering based on noun phrase dependency parsing and biomedical vocabularies

As a non-biomedical comparison we pool **fasttext** embeddings pre-trained on Wikipedia and Common Crawl.

The second category is sequence embedding, where words and/or sequences or words are embedded in context:

- **BioSentVec** sentence pooling: pass each sentence in the document to BioSentVec, then average pool the resulting embeddings³
- **BioBERT**: pool transformer outputs from the penultimate layer of Lee et al. (2019)’s fine-tuning of BERT on biomedical literature

As a non-biomedical comparison we pool the penultimate layer of **BERT-base**.

4 Tasks and Data

For quantifying document semantics, the ideal embedding is one where the vector space represents a conceptual or thematic space that is anchored to identifiable concepts and topics in the relevant domains. That is, we want documents with

similar embeddings to lie at similar points in a real-world conceptual space. Here we focus on two examples of such spaces: (1) MeSH terms (which include diseases, drugs, chemicals and many general topics and themes), and (2) at a coarser level of granularity, academic disciplines (e.g. cardiology, psychiatry).

MeSH headings provide a human-curated gold standard for medical publication semantics. Several approaches such as DeepMeSH (Peng et al. 2016) have been employed to try to solve the problem of automated MeSH indexing – many scientific articles lack MeSH annotations, either because they are not available on MedLine, or because they are too new to have been annotated. Rather than try to predict MeSH terms individually, we are using them as a gold-standard evaluation metric for our embedding methods, aimed at determining how well the vector space maps onto a known conceptual space in this domain.

To evaluate how well our document representations map onto the MeSH space, we vectorize the MeSH terms associated with a corpus of documents using inverse document frequency to penalize ubiquitous, less informative terms. Then, for random pairs of documents from that corpus, we correlate two metrics: (1) the cosine similarity of the MeSH vectors, and (2) the cosine similarity of our text-based vectors. For our corpus we randomly selected 10,000 pairs of documents (title + abstract) from MedLine. The intuition behind this intrinsic assessment is that the greater the correlation between the two similarity metrics being compared, the greater the extent to which those vectorizations encode the same conceptual space.

For our other task, we evaluate how well the methods do as text featurization methods for the task of learning to classify academic disciplines from document text. We have constructed a data set of over 2,000 recent faculty publications from the Zucker School of Medicine at Hofstra University, freely available at <http://academicworks.medicine.hofstra.edu>, along with the label of the department from which the publication originated. The department labels (e.g. cardiology, dermatology, neurology – 36 in all) serve as a proxy for academic

²In our initial tests we found that including bigrams slightly outperformed unigrams only across the board, and thus we only report these numbers.

³ Being based on sent2vec, BioSentVec is trained to embed sentences, and therefore performs slightly better when sentences are pooled, compared to when the entire title + abstract are embedded directly.

	TF-IDF	TF-IDF+	BioWordVec	BioWordVec+	BioSentVec	BioBERT	fasttext+	BERT-base
Linear Discriminant Analysis	0.26	0.32	0.54	0.56	0.39	0.39	0.47	0.38
Linear SVC	0.55	0.56	0.52	0.54	0.46	0.33	0.41	0.42
Multiclass Logistic Regression	0.55	0.55	0.49	0.54	0.49	0.34	0.37	0.42

Table 1: Results on department classification task (weighted F1).

TF-IDF	TF-IDF+	BioWordVec	BioWordVec+	BioSentVec	BioBERT	fasttext+	BERT-base
0.04	0.20	0.34	0.31	0.27	0.07	0.10	0.27

Table 2: MeSH correlation results (Spearman’s ρ)

discipline, a very coarse-grained measure of what the documents are about. The driving intuition is that the document embedding method that succeeds most in capturing document-level topical information should be better at separating out these classes. Each vectorization method serves as input to a number of classification models optimized on this task. The data set is relatively small, and thus the utility of transfer learning comes into play – we expect pre-trained models to succeed insofar as they add knowledge obtained from a much larger corpus. Moreover, if the densely embedded feature vectors succeed, they do so with much greater computational efficiency than Tf-idf, requiring learning from only a few hundred features, rather than over 100,000.⁴

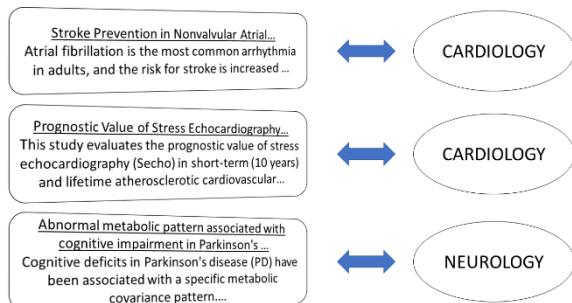


Figure 2: Example documents (left) and labels (right) from the Hofstra dataset.

5 Results

For the department prediction task, a number of classification layer architectures were tested using

each of the five document vectorization techniques outlined above, of which the best performing models – linear support vector classifier (Tang 2013), multiclass logistic regression and linear discriminant analysis – are reported. Hyperparameters were optimized separately for each model and input type using random search.

The results are shown in Table 1. We see that the lexical pooling methods generally outperform the sequence embedding methods, with the best results coming from a linear SVC on Tf-idf+ document vectors and linear discriminant analysis on BioWordVec+ document vectors. The non-biomedical fasttext model does not perform as well as BioWordVec, but surprisingly, BERT-base does outperform BioBERT in two of three cases.

Results of the MeSH correlation are given in Table 2. Here BioWordVec is the “winner”, i.e., these results suggest that these are the embeddings that best map onto the semantic space carved out by the expert-curated MeSH vocabulary. In this experiment, the ontology-based pre-filtering only introduced an advantage for the Tf-idf vectorizations. All correlations were statistically significant.

6 Discussion

Of the methods we compared, average pooling of biomedically trained word embeddings seems best suited to capture the document-level semantics of biomedical documents. BioWordVec+ performs similarly on the department classification task to its sparse Tf-idf counterpart, which performs surprisingly well and better than all the others. At the

⁴ We plan to make both of our data sets available to the public. For questions about access, please contact jon.stevens@abbvie.com

same time, BioWordVec pooling yields the closest approximation to MeSH-based document similarity. The broader implication of this is that meaningful embedded representations of biomedical abstracts can be obtained by a simple averaging of word vectors, and that in some cases, improvement can be found by using biomedical ontologies and noun phrase parsing filter out irrelevant words. Our results also reinforce the notion that domain matters – pooling fasttext vectors trained on large amounts of non-biomedical English does not produce as good a result. Document embeddings for scientific literature have numerous practical applications in the biomedical domain, because they are easily obtained, information-dense representations that can be stored in a database, quickly retrieved, and used in document classification models, search and text mining systems, and article recommender systems. Some mysteries remain to be addressed by future work, such as the underperformance of BioBERT, in particular when compared to the BERT-base model.

Acknowledgments

The authors would like to thank Brian Martin, Kevin Chiou, Mehmed Sariyildiz and the rest of the RAIDERS team, Rob Gregg, Sajeev Cherian, Masha Trenhaile, Kamron Mehrayin, and Kevyn Collins-Thompson.

References

- Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135-146.
- Chen, Q., Peng, Y., & Lu, Z. (2018). BioSentVec: creating sentence embeddings for biomedical texts. arXiv preprint arXiv:1810.09302.
- Chiu, B., Crichton, G., Korhonen, A., & Pyysalo, S. (2016). How to train good word embeddings for biomedical NLP. In Proceedings of the 15th workshop on biomedical natural language processing (pp. 166-174).
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
- Fleuren, W. W., & Alkema, W. (2015). Application of text mining in the biomedical domain. *Methods*, 74, 97-106.
- Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 328-339).
- Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In International conference on machine learning (pp. 1188-1196).
- Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2019). BioBERT: pre-trained biomedical language representation model for biomedical text mining. arXiv preprint arXiv:1901.08746.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).
- Pagliardini, M., Gupta, P., & Jaggi, M. (2018). Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (pp. 528-540).
- Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10), 1345-1359.
- Peng, S., You, R., Wang, H., Zhai, C., Mamitsuka, H., & Zhu, S. (2016). DeepMeSH: deep semantic representation for improving large-scale MeSH indexing. *Bioinformatics*, 32(12), i70-i79.
- Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).
- Tang, Y. (2013). Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).
- Wang, Y., Liu, S., Afzal, N., Rastegar-Mojarad, M., Wang, L., Shen, F., ... & Liu, H. (2018). A comparison of word embeddings for the biomedical natural language processing. *Journal of biomedical informatics*, 87, 12-20.
- Xiao, H. (2018). bert-as-service documentation. <https://github.com/hanxiao/bert-as-service>

Deep-EOS: General-Purpose Neural Networks for Sentence Boundary Detection

Stefan Schweter

Bayerische Staatsbibliothek München
Digital Library/Munich Digitization Center
Munich, Germany
`{stefan.schweter}@bsb-muenchen.de`

Sajawel Ahmed

Text Technology Lab
Goethe University Frankfurt
Frankfurt, Germany
`{sahmed}@em.uni-frankfurt.de`

Abstract

In this paper, we present three general-purpose neural network models for sentence boundary detection. We report on a series of experiments with long short-term memory (LSTM), bidirectional long short-term memory (BiLSTM) and convolutional neural network (CNN) for sentence boundary detection. We show that these neural networks architectures outperform the popular framework of *OpenNLP*, which is based on a maximum entropy model. Hereby, we achieve state-of-the-art results both on multi-lingual benchmarks for 12 different languages and on a *zero-shot* scenario, thus concluding that our trained models can be used for building a robust, language-independent sentence boundary detection system.

1 Introduction

The task of sentence boundary detection is to identify sentences within a text. Many natural language processing (NLP) tasks take a sentence as an input unit, such as part-of-speech tagging (Manning, 2011), dependency parsing (Yu and Vu, 2017), named entity recognition or machine translation. Thus, this foundational task stands at the beginning of various NLP processes and decisively determines their downstream-performance.

Sentence boundary detection is a nontrivial task, because of the ambiguity of the period sign “.”, which has several functions (Grefenstette and Tapanainen, 1994), e.g.:

- End of sentence
- Abbreviation
- Acronyms and initialism
- Mathematical numbers

A sentence boundary detection system has to resolve the use of ambiguous punctuation characters

to determine if the punctuation character is a true end-of-sentence marker¹.

In the present work, we train different deep architectures of neural networks, such as long short-term memory (LSTM), bidirectional long short-term memory (BiLSTM) and convolutional neural network (CNN), and compare the results with *OpenNLP*². *OpenNLP* is a state-of-the-art tool and uses a maximum entropy model for sentence boundary detection. To test the robustness of our models, we use the *Europarl* corpus for German and English, the *SETimes* corpus for nine different Balkan languages, and the *Leipzig* corpus (Goldhahn et al., 2012) for one Semitic language, namely Arabic. This makes our model language-independent, in which further languages can be used, given the associated training resources are available.

Additionally, we use a *zero-shot* scenario to test our model on unseen abbreviations. We show that our models outperform *OpenNLP* both for each language and on the zero-shot learning task. Therefore, we conclude that our trained models can be used for building a robust, language-independent state-of-the-art sentence boundary detection system.

The remainder of the paper is organized as follows: Section 2 reviews related work. Section 3 presents a sketch of the underlying neural models and the choice of hyperparameters. Section 4 describes the text data and its preprocessing for our twofold experimental setup of a) mono-lingual, and b) zero-shot training. Section 5 reports our results, and, finally, Section 6 discusses our results and draws a conclusion.

2 Related Work

Various approaches have been employed to achieve sentence boundary detection in different languages.

¹In this paper, we define “?!;,” as potential end-of sentence markers.

²*OpenNLP* 1.8.4: <https://opennlp.apache.org>

Recent research in sentence boundary detection focus on machine learning techniques, such as hidden Markov models (Mikheev, 2002), maximum entropy (Reynar and Ratnaparkhi, 1997), conditional random fields (Tomanek et al., 2007), decision tree (Wong et al., 2014) and neural networks (Palmer and Hearst, 1997). Kiss and Strunk (2006) use an unsupervised sentence detection system called *Punkt*, which does not depend on any additional resources. The system use collocation information as evidence from unannotated corpora to detect e.g. abbreviations or ordinal numbers.

The sentence boundary detection task can be treated as a classification problem. Our work is similar to the *SATZ* system, proposed by Palmer and Hearst (1997), which uses a fully-connected feed-forward neural network. The *SATZ* system disambiguates a punctuation mark given a context of k surrounding words. This is different to our approach, as we use a char-based context window instead of a word-based context window.

Further high-performers such as *Elephant* (Evang et al., 2013) or *Cutter* (Graen et al., 2018) follow a sequence labeling approach. However, they require a prior language-dependent tokenization of the input text. In contrast to these works, we construct an end-to-end approach which does not depend on the performance of any tokenization method, thus making our *Deep End-Of-Sentence detector* (Deep-EOS) more robust to multi-lingual settings.

3 Model

We use three different architectures of neural networks: long short-term memory (LSTM), bidirectional long short-term memory (BiLSTM) and convolutional neural network (CNN). All three models capture information at the character level. Our models disambiguate potential end-of-sentence markers followed by a whitespace or line break given a context of k surrounding characters. The potential end-of-sentence marker is also included in the context window. Table 1 shows an example of a sentence and its extracted contexts: left context, middle context and right context. We also include the whitespace or line break after a potential end-of-sentence marker.

LSTM We use a standard LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) network with an embedding size of 128. The number of hidden states is 256. We apply dropout with probability of 0.2 after the hidden layer during training. We apply a sigmoid non-linearity before the prediction layer.

Input sentence	Left	Middle	Right
I go to Mr. Pete Tong	to Mr	.	.Pete

Table 1: Example for input sentence and extracted context of window size 5.

bility of 0.2 after the hidden layer during training. We apply a sigmoid non-linearity before the prediction layer.

BiLSTM Our bidirectional LSTM network uses an embedding size of 128 and 256 hidden states. We apply dropout with a probability of 0.2 after the hidden layer during training, and we apply a sigmoid non-linearity before the prediction layer.

CNN For the convolutional neural network we use a 1D convolution layer with 6 filters and a stride size of 1 (Waibel et al., 1989). The output of the convolution filter is fed through a global max pooling layer and the pooling output is concatenated to represent the context. We apply one 250-dimensional hidden layer with ReLU non-linearity before the prediction layer. We apply dropout with a probability of 0.2 during training.

Other Hyperparameters Our proposed character-based model disambiguates a punctuation mark given a context of k surrounding characters. In our experiments we found that a context size of 5 surrounding characters gives the best results. We found that it is very important to include the end-of-sentence marker in the context, as this increases the F1-score of 2%. All models are trained with averaged stochastic gradient descent with a learning rate of 0.001 and mini-batch size of 32. We use Adam for first-order gradient-based optimization. We use binary cross-entropy as loss function. We do not tune hyperparameters for each language. Instead, we tune hyperparameters for one language (English) and use them across languages. Table 2 shows the number of trainable parameters for each model.

Model	# Parameters
LSTM	420,097
BiLSTM	814,593
CNN	33,751

Table 2: Number of trainable parameters for LSTM, bidirectional LSTM and CNN.

4 Experimental Setup

Data Similar to Wong et al. (2014) we use the *Europarl* corpus (Koehn, 2005) for our experiments. The *Europarl* parallel corpus is extracted from the proceedings of the European Parliament and is originally created for the research of statistical machine translation systems. We only use German and English from *Europarl*. Wong et al. (2014) does not mention that the *Europarl* corpus is not fully sentence-segmented. The *Europarl* corpus has a one-sentence per line data format. Unfortunately, in some cases one or more sentences appear in a line. Thus, we define the *Europarl* corpus as “quasi”-sentence segmented corpus. We use the *SETimes* corpus (Tyers and Alperen, 2010) as a second corpus for our experiments. The *SETimes* corpus is based on the content published on the *SETimes.com news portal* and contains parallel texts in ten languages. Aside from English the languages contained in the *SETimes* corpus fall into several linguistic groups: Turkic (Turkish), Slavic (Bulgarian, Croatian, Macedonian and Serbian), Hellenic (Greek), Romance (Romanian) and Albanic (Albanian). The *SETimes* corpus is also a “quasi”-sentence segmented corpus. For our experiments we use all the mentioned languages except English, as we use an English corpus from *Europarl*. We do not use any additional data like abbreviation lists. We use the *Leipzig* corpus as the third and final corpus to include the non-European language Arabic into the scope of our investigations. For a *zero-shot* scenario we extracted 80 German abbreviations including their context in a sentence from Wikipedia. These abbreviations do not exist in the German *Europarl* corpus.

Preprocessing All corpora are not tokenized. Text tokenization (or, equivalently, segmentation) is highly non-trivial for many languages (Schütze, 2017). It is problematic even for English as word tokenizers are either manually designed or trained. For our proposed sentence boundary detection system we use a similar idea from Lee et al. (2017). They use a character-based approach without explicit segmentation for neural machine translation. We also use a character-based context window, so no explicit segmentation of input text is necessary.

For all corpora we use the following preprocessing steps: (a) we remove duplicate sentences, (b) we extract only sentences with ends with a potential end-of-sentence marker. Each text collection

Language	# Train	# Dev	# Test
German	1,476,653	184,580	184,580
English	1,474,819	184,352	184,351
Arabic	1,647,906	274,737	276,172
Bulgarian	148,919	18,615	18,614
Bosnian	97,080	12,135	12,134
Greek	159,000	19,875	19,874
Croatian	143,817	17,977	17,976
Macedonian	144,631	18,079	18,078
Romanian	148,924	18,615	18,615
Albanian	159,323	19,915	19,915
Serbian	158,507	19,813	19,812
Turkish	144,585	18,073	18,072

Table 3: Number of sentences in *Europarl*, *SETimes* and *Leipzig* corpus for each language for training, development and test set.

for a language is split into train, dev and test sets. Table 3 shows a detailed summary of the training, development and test sets used for each language.

Tasks In the first task we train our different models on the *Europarl*, *SETimes* and *Leipzig* corpus. The second task is to perform *zero-shot* sentence boundary detection. For the *zero-shot* scenario the trained models for the German *Europarl* corpus are used.

Setup We evaluate our different models on our three corpora. We measure F1-score for each model. As baseline to our models, we use *OpenNLP*. *OpenNLP* uses a maximum entropy model. *OpenNLP* comes with pretrained models for German and English, but to ensure a fair comparison between our models and *OpenNLP*, we do not use them. Instead, we train a model from scratch for each language with the recommended hyperparameters from the documentation. For the *zero-shot* scenario we use our trained LSTM, BiLSTM and CNN models on the German *Europarl* corpus and the trained model with *OpenNLP* to perform a zero-shot sentence boundary detection on the crawled abbreviations.

5 Results

We train a maximum of 10 epochs for each model. For the German and English corpus (*Europarl*) the time per epoch is 55 minutes for the BiLSTM model, 28 minutes for the LSTM model and 5 minutes for the CNN model. For each language from the *SETimes* corpus the time per epoch is 5 minutes

Lang.	LSTM	BiLSTM	CNN	OP
German	97.59	97.59	97.50	97.38
English	98.61	98.62	98.55	98.40
Arabic	99.86	99.83	81.97	99.76
Bulg.	99.22	99.27	99.22	98.87
Bosn.	99.58	99.52	99.53	99.25
Greek	99.67	99.70	99.66	99.25
Croat.	99.46	99.44	99.44	99.07
Maced.	98.04	98.09	97.94	97.86
Roman.	99.05	99.05	99.06	98.89
Alban.	99.52	99.51	99.47	99.34
Serbian	98.72	98.76	98.73	98.32
Turkish	98.56	98.58	98.54	98.08

Table 4: Results on test set for *Europarl*, *SETimes* and *Leipzig* corpus against *OpenNLP* (OP). The highest F1-score for each task on each language is marked in bold face.

for the Bi-LSMT model, 3 minutes for the LSTM model and 20 seconds for the CNN model. Timings are performed on a server machine with a single Nvidia Tesla K20Xm and Intel Xeon E5-2630.

The results on test set on the *SETimes* corpus are shown in Table 4. For each language the best neural network model outperforms *OpenNLP*. On average, the best neural network model is 0.38% better than *OpenNLP*. The worst neural network model also outperforms *OpenNLP* for each language. On average, the worst neural network model is 0.33% better than *OpenNLP*. In half of the cases the bi-directional LSTM model is the best model. In almost all cases the CNN model performs worse than the LSTM and bi-directional LSTM model, but it still achieves better results than the *OpenNLP* model. This suggests that the CNN model still needs more hyperparameter tuning.

The first two rows in Table 4 show the results on test set on the *Europarl* corpus. For both German and English the best neural network model outperforms *OpenNLP*. The CNN model performs worse than the LSTM and bi-directional LSTM model but still achieves better results than *OpenNLP*. The bi-directional LSTM model is the best model and achieves the best results for German and English. On average, the best neural network model is 0.22% better than *OpenNLP*, whereas the worst neural network model is still 0.14% better than *OpenNLP*.

Table 5 shows the results for the *zero-shot* scenario. The CNN model outperforms *OpenNLP* by

Model	Precision	Recall	F1
LSTM	56.62	96.25	71.29
BiLSTM	60.00	97.50	74.29
CNN	61.90	97.50	75.12
<i>OpenNLP</i>	54.60	96.25	69.68

Table 5: Results on the *zero-shot* scenario for unseen German abbreviations.

a large margin and is 6% better than *OpenNLP*. The CNN model also outperforms all other neural network models. Interestingly, the CNN model performs better in a *zero-shot* scenario than in the previous tasks (*Europarl* and *SETimes*). That suggests that the CNN model generalizes better than LSTM or BiLSTM for unseen abbreviations. The worst neural network model (LSTM model) still performs 1,6% better than *OpenNLP*.

6 Discussion & Conclusion

In this paper, we propose a general-purpose system for sentence boundary detection using different architectures of neural networks. We use the *Europarl*, *SETimes* and *Leipzig* corpus and compare our proposed models with *OpenNLP*. We achieve state-of-the-art results.

The results on the three corpora show that the trained neural network models perform well for all languages. We tune hyperparameters just for one language (English) and share these hyperparameter settings across other languages. This suggests that the proposed neural network models can adopt other languages as well, which makes them language-independent. Our character-based context approach requires no explicit text segmentation and is robust against unknown words.

In a *zero-shot* scenario, in which no manifestation of the test abbreviations is observed during training, our system is also robust against unseen abbreviations. It shows that our proposed neural network models can detect abbreviations “on the fly”, after the model has already been trained.

The fact that our proposed neural network models perform well on different languages and on a *zero-shot* scenario leads us to the conclusion that *Deep-EOS* is a *general-purpose* system³. Our system can be used for a wide variety of practical use cases, e.g. in the scope of the *BIOfid* project where unstructured OCR text data on biodiversity has to

³<https://github.com/stefan-it/deep-eos>

be processed for the task of biological Named Entity Recognition (Ahmed and Mehler, 2018; Ahmed et al., 2019).

Acknowledgments

We would like to thank the *Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften* (LRZ) for giving us access to the NVIDIA DGX-1 supercomputer. Special thanks go to Prof. R. V. Zicari and Prof. A. Mehler for their constructive comments on the final manuscript, to Prof. J. Leidner for his remarks on the related work, and last but not least to Prof. U. Meyer and *Goethe Research Academy for Early Career Researchers* (GRADE) for funding the publication process of this paper.

References

- Sajawel Ahmed and Alexander Mehler. 2018. Resource-Size matters: Improving Neural Named Entity Recognition with Optimized Large Corpora. In *Proceedings of the 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*.
- Sajawel Ahmed, Manuel Stoeckel, Christine Driller, Adrian Pachzelt, and Alexander Mehler. 2019. BIOfid Dataset: Publishing a German Gold Standard for Named Entity Recognition in Historical Biodiversity Literature. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. Association for Computational Linguistics. accepted.
- Kilian Evang, Valerio Basile, Grzegorz Chrupała, and Johan Bos. 2013. Elephant: Sequence labeling for word and sentence segmentation. In *EMNLP 2013*.
- Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. 2000. Learning to Forget: Continual Prediction with LSTM. *Neural Comput.*, 12(10):2451–2471, October.
- Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In *LREC*.
- Johannes Graën, Mara Bertamini, and Martin Volk. 2018. Cutter—a Universal Multilingual Tokenizer. In *Proceedings of the 3rd Swiss Text Analytics Conference-SwissText*, pages 75–81.
- Gregory Grefenstette and Pasi Tapanainen. 1994. What is a word, What is a sentence? Problems of Tokenization. In *COMPLEX*, pages 79–87.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November.
- Tibor Kiss and Jan Strunk. 2006. Unsupervised Multilingual Sentence Boundary Detection. *Comput. Linguist.*, 32(4):485–525, December.
- Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.
- Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *TACL*, 5:365–378.
- Christopher D. Manning. 2011. Part-of-speech Tagging from 97% to 100%: Is It Time for Some Linguistics? In *CICLing*, pages 171–189, Berlin, Heidelberg. Springer-Verlag.
- Andrei Mikheev. 2002. Periods, Capitalized Words, etc. *Comput. Linguist.*, 28(3):289–318, September.
- David D. Palmer and Marti A. Hearst. 1997. Adaptive Multilingual Sentence Boundary Disambiguation. *Comput. Linguist.*, 23(2):241–267, June.
- Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A Maximum Entropy Approach to Identifying Sentence Boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing, ANLC ’97*, pages 16–19, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Hinrich Schütze. 2017. Nonsymbolic Text Representation. In *EACL*, pages 785–796.
- Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. Sentence and token splitting based on conditional random fields. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 49–57.
- Francis M Tyers and Murat Serdar Alperen. 2010. South-east european times: A parallel corpus of balkan languages. In *Proceedings of the LREC Workshop on Exploitation of Multilingual Resources and Tools for Central and (South-) Eastern European Languages*, pages 49–53.
- Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. 1989. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339.
- Derek F Wong, Lidia S Chao, and Xiaodong Zeng. 2014. iSentenizer- μ : Multilingual sentence boundary detection model. *The Scientific World Journal*, 2014.
- Xiang Yu and Ngoc Thang Vu. 2017. Character Composition Model with Convolutional Neural Networks for Dependency Parsing on Morphologically Rich Languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 672–678, Vancouver, Canada, July. Association for Computational Linguistics.

How Does Visual Complexity Influence Predictive Language Processing in a Situated Context?

Özge Alaçam, Wolfgang Menzel and Tobias Staron

Department of Informatics

University of Hamburg, Hamburg 22527, Germany

{alacam, menzel, staron}@informatik.uni-hamburg.de

Abstract

In this study, by using a visual world paradigm, we investigate to what extent predictive language processing is affected by the visual complexity of the scene. The visual complexity is manipulated by adding irrelevant background objects or irrelevant characters (w.r.t. the spoken utterance) to the scene. The results of the eye-tracking experiment, that point out significant effects of visual complexity on situated-language processing, provide basic insights for designing cross-modal language understanding systems.

Keywords: incremental / predictive language processing; visual complexity

1 Predictive Language Processing

A large body of empirical evidence in psycholinguistics indicates that the interaction between linguistic and visual modalities plays a crucial role in predicting what will be revealed next in the unfolding sentence (Altmann and Kamide, 1999; Knoeferle, 2005; Tanenhaus et al., 1995).

Altmann and Kamide (1999)'s study on structural prediction has documented that listeners are able to predict complements of a verb based on its selectional constraints and immediately begin incremental parsing operations. For example, when people hear the verb "break", their attention is directed towards only breakable objects in the scene.

Focusing on prosodic cues and visual saliency, Coco and Keller's research (2015) goes deeper into the understanding of which kinds of information play role on different comprehension processes regarding situated predictive language processing. This study contains three systematic manipulations; namely (i) only the visual saliency, (ii) only the linguistic saliency and (iii) both of them together. The results point out that visual saliency narrows down

the visual search space towards a target, but does not have a direct role on linguistic ambiguity resolution, while different intonational breaks favor one interpretation over the other. On the other hand no statistical interaction effect between the two modalities has been found although they complement each other and both contribute to the overall understanding of the sentence by having different roles.

Numerous studies on structural prediction have only been carried out on relatively simple visual and linguistic settings where object-action relations could be predicted relatively easily, except several studies (Ferreira et al., 2013; Coco and Keller, 2015). Therefore, to what extent this multimodal interaction occurs still needs extensive systematic investigation. As reported by Ferreira et al. (2013), when the visual context is complex, subjects have difficulties using visual information to narrow down their hypotheses about possible interpretations. Depending on the complexity of the visual environment or task, humans might choose a more passive strategy (such as waiting to have complete information about the entities referred to in the utterance) instead of anticipating upcoming information, and humans have such a preference for the cases where there is a high risk of faulty prediction.

Our project focuses on studying underlying mechanisms of human cross-modal language processing of incrementally revealed utterances with accompanying visual scenes, with the aim of using the empirically gained insights to develop a fluent and efficient cross-modal and incremental parsing solutions. Better understanding of human perceptual and comprehension processes is one of the crucial factors in the realization of dynamic human-computer interaction. This paper addresses the empirical aspects of human language processing particularly focusing on the influence of different irrelevant visual components on the predictive lan-

guage processing.

2 Experiment

One of the well-investigated phenomena regarding the interplay between language and vision is the *garden-path effect*, which occurs when the prediction made for the upcoming sentence part does not match with the real situation, requiring re-analysis of the interpretation during online language comprehension. Subject-object ambiguities in German elicit garden-path effects due to sometimes ambiguous case marking. This phenomenon has already been observed by Knoeferle et al. (2005). In their study, two different sentence patterns are compared; unmarked word order ($\text{Subject}_{\text{ambiguous}} \text{ Verb } \text{Object}_{\text{accusative}}$) and marked word order ($\text{Object}_{\text{ambiguous}} \text{ Verb } \text{Subject}_{\text{nominative}}$). Each sentence addresses only one action between two characters and is accompanied by a scene that depicts two actions and three characters (one ambiguous AGENT/PATIENT character, one PATIENT, and one AGENT). If the preference-driven initial role assignment for the first noun phrase (before the verb) creates a conflict with the late assignment for the second one (following the verb), a reanalysis becomes necessary. Eye-tracking results show that this happens only in case of the marked word order. More interestingly, visual attention already starts to move towards the target character before the associated post-verbal noun phrase actually becomes available, i.e. while the verb is still being spoken. This clearly signals that reference resolution for the second noun phrase is not based on the observation of the phrase itself but on its prediction induced by the verb. To find out whether this effect also occurs in more complex visual environments, we compiled a set of four different visual conditions accompanied by two different versions of AGENT/PATIENT order following the same sentence patterns.

[1] Unmarked word order (SVO):

Die Arbeiterin(*f, nom*) kostümiert mal eben den jungen Mann(*m, acc*).
The worker just dresses up the young man.

[2] Marked word order (OVS):

Die Arbeiterin(*f, acc*) verköstigt mal eben der Astronaut(*m, nom*).
The worker is just fed¹ by the astronaut.

¹The original German sentence is in active voice in OVS word order.

The spoken sentences were recorded by a male native speaker of German at a normal speech rate. We avoided unequal intonational breaks that may bias the interpretation.

The visual context differs from those of Knoeferle et al. (2005) by a more realistic and complex background. In particular, the pictures contain more information than the sentences describe. We manipulate the visual complexity in four different conditions, see Figure 1. In the first condition (C1), the scene contains three characters (one ambiguous AGENT/PATIENT candidate, one for the PATIENT role, and one for AGENT) as in the original study. In the second condition (C2), more background objects are added, while in the third condition (C3) an additional character is included. The distractor objects and the characters have no direct thematic relation regarding the spoken sentence, therefore they should not affect the fixations on the target object particularly in the unmarked sentences. However, the additional character is acting on the ambiguous AGENT/PATIENT character, thus increasing the complexity of referential selection. Therefore, we hypothesize that having more background objects but no additional character (C2) should distract the subjects from the target not as much as condition (C3) does. Finally, C4 is a combination of C2 and C3.

In order to ensure the compatibility of the verbs and nouns with their depiction in the scenes, the scenes (with background objects and four characters) were shown to 15 participants, 4 of 40 stimuli were excluded from the stimuli set, since at least one of the actions/nouns was found not easily depictable.

27 university students participated in the experiment. The experiment was conducted in German by native speakers. Using the visual-world paradigm, a total of 32 visual displays with accompanying spoken utterances were utilized. The stimuli were displayed on an SR Eyelink 1000 Plus eye tracker with a sampling rate of 1000 Hz. The 2D visual scenes were created with the SketchUp Make Software² with a resolution of 1250 x 947px.

A visual scene is presented for 10 sec before the onset of the spoken sentence. The preview gives a comprehender time to encode the visual information so visual attention will be free of recognizing the objects during language processing. Then, the spoken sentence is presented accompanying the vi-

²<http://www.sketchup.com/> – retrieved on 10.09.2018

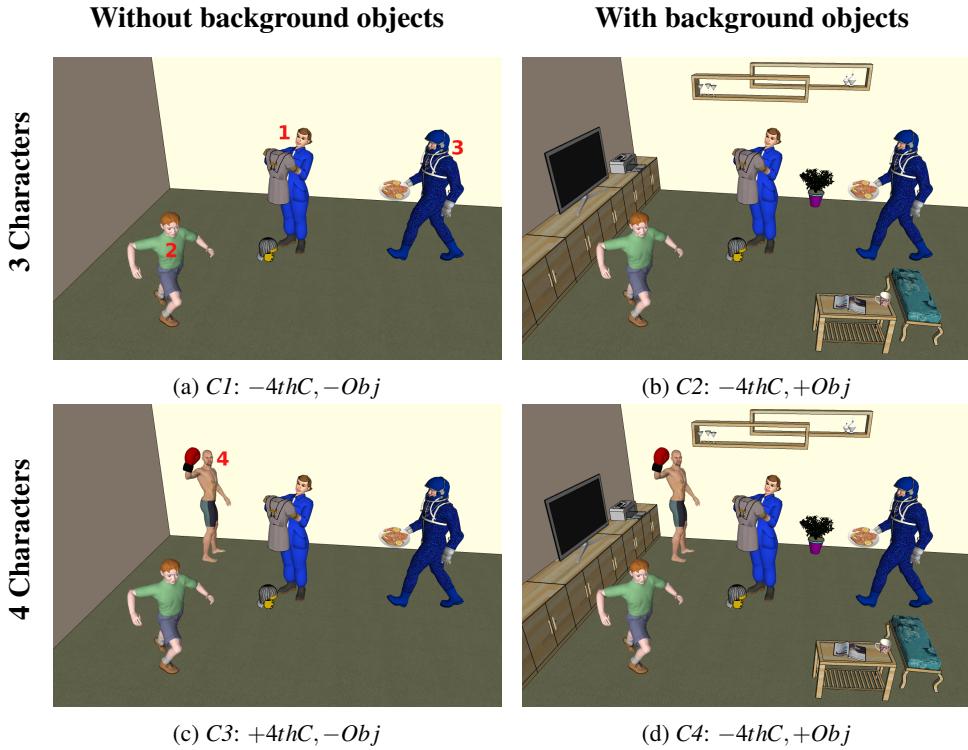


Figure 1: Example stimuli in four different visual complexity conditions

sual stimulus. Participants are asked to examine the scene carefully and attend the information given in the audio. The order of stimuli is randomized for each participant.

In this study, we focus only on the time course of fixations on the target (either AGENT or PATIENT depending on the word order) and on the competitor characters after the onset of the verb until its offset. Due to varying word length, the time window for a verb duration was normalized by stretching each individual time series to the maximum verb length observed and then reduced to 31 bins by aggregating 5 bins to one. The time window is shifted forward 200 ms in order to account for the time required to initiate eye movement (Matin et al., 1993). In total, fixation distributions of 858 trials (27 participant * 32 scenes) per character were evaluated. The fixations were coded as binomial w.r.t. whether the character is fixated or not. Fixation parameter was transformed into empirical logitbased on *population-average* estimates with weightsfollowing the reasoning discussed in (Barr, 2008; Jaeger, 2008).

3 Results

All analyses were carried out in R version 3.5.1. (Team, 2013) by utilizing *Lmertest*, *Lme4* and

multcomp packages. Due to the expected curvilinear change over time, a higher-order polynomial model was chosen (Mirman, 2016; Baayen et al., 2008) to analyze the effects of word order, and visual complexities over the course of unfolding sentence. We start out with a "base" model of fixation distribution over time with crossed-random effects of items and subjects on all orthogonal polynomial terms. Adding the visual complexity parameter (*Vis*) significantly improved model fit ($\chi^2(3) = 182.75, p < .001$), as well as the word order parameter (*Ling*), ($\chi^2(1) = 36.26, p < .001$). Finally, the full model with all interaction effects of the fixed terms provided the best fit compared to previous models ($\chi^2(15) = 498.78, p < .001$).

As given in Table 1, the main effect of word order is significant indicating that the fixation distribution in two word order conditions differs significantly. There is also a significant effect of word order on the linear term for both characters, indicating that the slope of the fixation distribution is steeper in the OVS compared to that in the SVO order. Figure 2 shows that – when the sentence in SVO order unfolds – the look on the target objects increases, and the look on the competitor decreases (blue lines). However, in the OVS order, an increase is observed both on the target and competitor characters (orange lines).

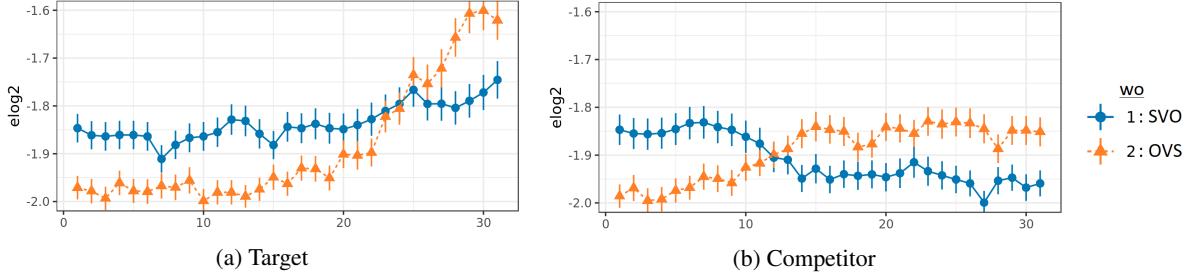


Figure 2: Time course of fixations on the target and competitor characters in two word order conditions

Target			Competitor			
Est.	SE	p	Est.	SE	p	
intercept	-.19	.02	< .001*	.27	.02	< .001*
linear	.99	.08	< .001*	.75	.09	< .001*
quadratic	.15	.08	.07	-.61	.09	< .001*

Table 1: Estimated parameters for the word order parameter on the target and competitor characters

Contrasts	Linear			Quadratic		
	Est.	SE	p	Est.	SE	p
1 vs. 2	-.48	.06	< .001*	-.51	.06	< 0.001*
1 vs. 3	-.34	.06	< .001*	-.19	.06	< 0.001*
1 vs. 4	-.48	.06	< .001*	-.25	.06	< 0.001*
2 vs. 3	-.15	.06	< .05*	.32	.06	> .05
3 vs. 4	-.14	.06	< .05*	-.06	.06	> 0.05

Table 2: Multiple comparisons on the target character regarding four visual complexity levels

As summarized in Table 2, the first three contrasts on slope (linear term) and curvature (quadratic term) of the fixation parameter over time show that the slope of the *C1* is always steeper with significantly more fixations compared to the other three complexities. Figure 3 shows that the look on the target character starts around timestamp 21 (out of 31), directly after half of the verb has been uttered. On the other hand, this reaction is significantly slower for the other conditions. When background objects are added (*C2*), less fixation is observed on the target characters. The inclusion of the 4th character *C3* also causes a decrease of the look on the target compared to the first condition. The same pattern is observed between *C1* and *C4*.

The difference between *C2* and *C3* indicates that although the curvatures of the fixation distributions display similar pattern (quadratic term is > 0.05), the slope terms are significantly different indicating that adding 4th character results in overall slower increase on the fixation to the target compared to adding background objects *C2*. Finally, 3rd and 4th condition also show a difference only in the slope

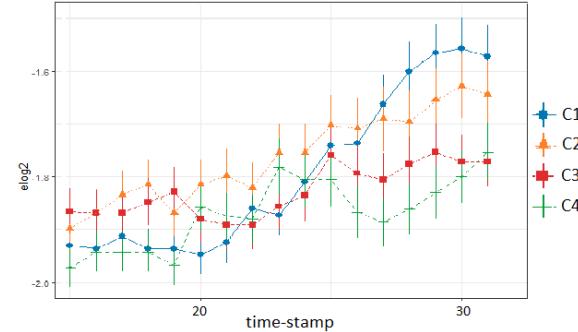


Figure 3: Time course of fixations on the target character for four visual complexity levels

term but not in the curvature.

Further analysis on the interaction between the word order and the visual complexity parameters indicated that while the fixation distributions for the sentences in different word order conditions significantly differed within the first (*Estimate* = -0.133 , *SE* = 0.013 , $p < 0.001$) and the second (*Estimate* = -0.062 , *SE* = 0.012 , $p < 0.001$) complexity levels, it, however, seems that this difference between the unmarked and marked sentence forms starts to fade with the inclusion of additional character (non-significant differences in SVO and OVS fixations for the 3rd and 4th visual complexity levels).

4 Discussion and Conclusion

The results replicate the findings previously reported in the literature that states the participants are garden-pathed when they hear a sentence in a OVS order, in which the expected order of the thematic roles are reversed (Knoeferle, 2005). This is in line with the NVN strategy, which states there is a tendency to assume that the first argument of a sentence is a proto-agent and the second is a proto-patient (Ferreira, 2003). Moreover, in OVS order one can see a late increase on the target. This result is also in line with the surprisal effect theory that

states that less predictable items are fixated longer (Levy, 2008; Hale, 2001). However, when we look at the comparisons in more detail for each visual complexity level separately, it seems that while the difference between the unmarked and marked sentences is preserved for the simple visual complexity conditions, no clear sign of prediction of the upcoming sentence part is observed when the complexity is increased. Similar to the findings by Coco and Keller (2015), this result implies that visual complexity may not have direct role on thematic role assignment, however, it does definitely have an effect on target identification.

Furthermore, our results also support an interactive processing architecture that claims visual information influences the processing of syntactic linguistic information (MacDonald and Seidenberg, 2006), in our case even when they are irrelevant to sentence context. Regarding visual complexity, although none of the manipulations directly has an association with the entities mentioned in the sentence, the results still reveal that the looks on the target are affected by the complexity of the environment, probably due to visual search despite given 10s preview. In the C1 condition, people look at the target object more compared to other conditions. The overall fixation rate decreases when the complexity increases (also confirming that having 4th character distracts more compared to having background objects).

Although our investigations into this area are still ongoing, the results could be a useful aid for developing models for cross-modal NLP that aim to account for visual complexity. The most interesting outcome for NLP implementation is to exhibit the varying effect of different and irrelevant visual objects on language processing. This highlights the fact that while contributing visual information into language processing, the effect of different visual components should be treated differently with respect to not just their direct relevance but also possible interference even though they do not have a direct relation to the linguistic input.

In a further study, we aim to address how our context-integrated parser reacts to those visual manipulations and whether thematic role assignments are affected by irrelevant and varying visual components.

Acknowledgments

This research was funded by the German Research Foundation (DFG) in the project Cross-modal Learning, TRR-169.

References

- Altmann, G., and Kamide, Y. 1999. Incremental interpretation at verbs: Restricting the domain of subsequent reference. *Cognition*, 73(3):247–264.
- Baayen, R., Davidson, D J., and Bates, D M. 2008. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of memory and language*, 59(4):390–412.
- Bailey, K., and Ferreira, F. 2007. The processing of filled pause disfluencies in the visual world. *Eye movements: A window on mind and brain*, 485–500.
- Knoeferle, P S. 2005. The role of visual scenes in spoken language comprehension: Evidence from eye-tracking. *The role of visual scenes in spoken language comprehension: Evidence from eye-tracking*. Universitätsbibliothek.
- Coco, M I., and Keller, F. 2015. The interaction of visual and linguistic saliency during syntactic ambiguity resolution. *The Quarterly Journal of Experimental Psychology*, 68(1):46–74.
- Ferreira, F. 2003. The misinterpretation of noncanonical sentences. *Cognitive psychology*, 47(2):164–203.
- Van Berkum, J J., Brown, C M., Zwitserlood, P., Kooijman, V., and Hagoort, P. 2005. Anticipating upcoming words in discourse: evidence from ERPs and reading times. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 31(3):443.
- Ferreira, F., Foucart, A., and Engelhardt, P E. 2013. Language processing in the visual world: Effects of preview, visual complexity, and prediction. *Journal of Memory and Language*, 69(3):165–182.
- Matin, E., Shao, K C., and Boff, K R. 1993. Saccadic overhead: Information-processing time with and without saccades. *Perception & psychophysics*, 53(4):372–380.
- Barr, D J. 2008. Analyzing ‘visual world’ eye-tracking data using multilevel logistic regression. *Journal of memory and language*, 59(4):457–474.
- Jaeger, T F. 2008. Categorical data analysis: Away from ANOVAs (transformation or not) and towards logit mixed models. *Journal of memory and language*, 59(4):434–446.
- Meng, M., and Bader, M. 2000. Mode of disambiguation and garden-path strength: An investigation of subject-object ambiguities in German. *Language and Speech*, 43(1):43–74.

Mirman. D. 2016. Growth curve analysis and visualization using R *CRC Press*.

Snedeker. J., and Trueswell. J. 2003. Using prosody to avoid ambiguity: Effects of speaker awareness and referential context *Journal of Memory and language*, 48(1) 103–130.

Tanenhaus. M K., Spivey-Knowlton. M J., Eberhard. K M., and Sedivy. J C. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268(5217):1632.

Team. R C. 2013. R: A language and environment for statistical computing

Bates. D. 2005. Fitting linear mixed models in R. *R news*, 5(1): 27–30.

Hale. J. 2001. A probabilistic Earley parser as a psycholinguistic model. *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, 1–8.

Levy. R. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

MacDonald. M C., and Seidenberg. M S. 2006. *Handbook of psycholinguistics*.

McRae. K., Hare. M., Ferretti. T., and Elman. J L. 2001. Activating verbs from typical agents, patients, instruments, and locations via event schemas. *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society*, 617–622.

Quené. H. and Van den Bergh. H. 2008. Examples of mixed-effects modeling with crossed random effects and with binomial data. *Journal of Memory and Language*, 59(4):413–425.

Teacher-Student Learning Paradigm for Tri-training: An Efficient Method for Unlabeled Data Exploitation

Yash Bhalgat

Qualcomm AI Research

yashbhalgat95@gmail.com

Zhe Liu, Pritam Gundecha, Jalal Mahmud, Amita Misra

IBM-Research, Almaden

{liuzh, psgundec, jumahmud, amita.misra1}@us.ibm.com

Abstract

Given that labeled data is expensive to obtain in real-world scenarios, many semi-supervised algorithms have explored the task of exploitation of unlabeled data. Traditional tri-training algorithm and tri-training with disagreement have shown promise in tasks where labeled data is limited. In this work, we introduce a new paradigm for tri-training, mimicking the real world teacher-student learning process. We show that the adaptive teacher-student thresholds used in the proposed method provide more control over the learning process with higher label quality. We perform evaluation on SemEval sentiment analysis task and provide comprehensive comparisons over experimental settings containing varied labeled versus unlabeled data rates. Experimental results show that our method outperforms other strong semi-supervised baselines, while requiring less number of labeled training samples.

1 Introduction

Machine learning algorithms often require large amount of labeled data for training. As collecting labeled examples can be expensive, semi-supervised learning has been proposed (Zhu, 2006). Among the existing semi-supervised approaches, self-training (Triguero et al., 2015), co-training (Blum and Mitchell, 1998), and tri-training (Zhou and Li, 2005) are the most notable ones. However, they suffer from one major issue of the gradually increased level of noise during the iterative labeling process. This problem can be attributed to two factors: (1) static labeling threshold, and (2) inappropriate stopping criteria.

Many self-labeled algorithms iteratively enlarge labeled training set with unlabeled instances whose

prediction confidence is larger than a static labeling threshold. Static labeling threshold produces a good classification performance only when the proportion of correctly labeled instances remains above a constant level. However, given the continuously added noisy labels during the semi-supervised process (Triguero et al., 2015), it is unlikely that any fixed assignment of the threshold will produce optimal classifications.

Besides, deciding when to stop the iterative instance labeling process is also critical for the self-labeled techniques. Existing stopping criteria include: setting a threshold on the number of labels that the algorithm is willing to generate, or stopping the labeling process when little to no accuracy increase occurs in an iteration. Stopping criteria is still an open issue, as too conservative or too liberal stopping criteria may produce many mislabeled examples to the self-labeled process.

To solve the two challenges, we propose a new tri-training-based method, called tri-training with teacher-student paradigm. Specifically, in each iteration, a double-teacher-single-student teaching relation is established based on predefined teacher and student thresholds, where teachers teach the student with generated proxy labels on the unlabelled data. Along the teaching process, the teacher-student relationship is continuously adjusted with adaptive teacher and student thresholds. The teacher-student relationship terminates on either running out of teachable instances or when reaching a *graduation point*, where the student threshold equals the teacher threshold.

We evaluate the tri-training with teacher-student paradigm approach on the sentiment analysis task of SemEval-2016 over various labeled-unlabeled data ratios. The proposed method outperforms many strong baselines in terms of gaining better prediction performances while consuming less number of unlabeled examples.

2 Method

Assume we are given a set of unlabeled samples U as well as a set of labeled samples L , where $L \ll U$. The proposed method starts by training three independent base classifiers m_i, m_j, m_k on bootstrapped sample subsets S_i, S_j, S_k respectively taken from L . The aim of the bootstrap sampling is to increase the diversity of base classifiers trained through the labeled set. Next, for every sample x in U , each of the trained models m_i, m_j, m_k predicts a label c_i, c_j, c_k with corresponding prediction probability $p_i(c_i|x), p_j(c_j|x), p_k(c_k|x)$.

2.1 Teacher-Student Assignment

Instead of assigning x a majority voted label, as implemented in the original tri-training (Zhou and Li, 2005), here we model the learning task from a teacher-student perspective. In each iteration of our proposed approach, two classifiers (m_j and m_k) are ascertained to be teachers if their prediction probabilities $p_j(c_j|x)$ and $p_k(c_k|x)$ are both larger than the teacher threshold τ_t . The other classifier m_i is then treated as student if its prediction probability is less than the student threshold τ_s . An unlabeled sample x in U will only be assigned a label after it is identified as *teachable*. Teachable examples are defined according to the function *SelectTeachableSamples*, as shown in Algorithm 2. The required criteria are as follows: Firstly, the predicted labels c_j and c_k from the two teachers m_j and m_k must agree with each other. Second, both teachers' prediction confidences p_j and p_k must exceed τ_t and at the same time, the student's confidence p_i must be less than τ_s . This setting of using two teachers ensures that bias in any of these models doesn't affect the quality of the information taught to the student. It's similar to the real-life teacher-student learning process, where only qualified teachers can teach students things that they are the most comfortable with. Here, it is important to note that the teacher-student roles are rotated in each iteration, $i \in \{1, 2, 3\}, (j, k \neq i)$, allowing each classifier to learn from the other classifiers' experiences, as m_i is further trained with the original labeled set L along with the identified teachable samples L_i .

2.2 Adaptive Thresholds

Another novel aspect that we adopt from real-world teaching scenarios to the proposed method is the continuously adjusted teacher-student relationship. To be more specific, as a student learns from the

Algorithm 1 Teacher Student Tri-training

Require: L - set of labeled samples, U - set of unlabeled samples, $m_{i,j,k}$ - teacher-student models, τ_t - teacher threshold, τ_s - student threshold, λ_t, λ_s - teacher-student adaptive rates

```

1: for  $i \in \{1..3\}$  do
2:    $S_i \leftarrow \text{bootstrap\_sample}(L)$ 
3:    $m_i \leftarrow \text{train\_model}(S_i)$ 
4: end for
5: while  $\tau_s \leq \tau_t$  do
6:   for  $i \in \{1..3\}$  do
7:      $L_i \leftarrow \text{SelectTeachableSamples}(U, m_{i,j,k}, \tau_t, \tau_s)$ 
8:      $m_i \leftarrow \text{train\_model}(L \cup L_i)$ 
9:   end for
10:   $\tau_t \leftarrow \tau_t - \lambda_t$ 
11:   $\tau_s \leftarrow \tau_s + \lambda_s$ 
12: end while
13: apply majority vote over  $m_{i,j,k}$ 
```

Algorithm 2 Select Teachable Samples

Require: U - set of unlabeled samples, τ_t - teacher threshold, τ_s - student threshold, m_i - student model, $m_{j,k}$ - teacher models

```

1:  $\pi \leftarrow \emptyset$ 
2: for all  $x \in U$  do
3:   if  $c_j = c_k$  then
4:      $tcf = \min(p_j(c_j|x), p_k(c_k|x))$ 
5:      $scf = p_i(c_j|x)$ 
6:     if  $tcf > \tau_t$  &  $scf < \tau_s$  then
7:        $\pi \leftarrow \pi \cup \{(x, c_j(x))\}$ 
8:     end if
9:   end if
10: end for
11: return  $\pi$ 
```

teachers, it would become more confident of its prior knowledge taught by the teachers. In that sense, the student threshold τ_s increases monotonically in every iteration. On the other hand, as student progresses through the learning process, the teachers are supposed to teach them more advanced cases, i.e. cases where the teachers are less confident about. This is captured in our approach by monotonically decreasing the teacher threshold τ_t . For this work, we chose a linear adaptive rate for the adaptive process as shown in line 10 and 11 of Algorithm 1.

2.3 Stopping Criteria

Existing self-labeled techniques often stop when no sample can be labeled, or no performance improvement occurs in an iteration. The original tri-training paper introduces an error constraint that checks if a peak performance has been reached. However, the error measurement is conducted only on the labeled dataset, hence assuming that the labeled set distribution is representative of the unlabeled set distribution. Tri-training may also lead to a limited number of co-labeling examples for training and

a premature termination while dealing with large datasets (Chou et al., 2016).

In this work, we present our stopping criterion by comparing the student’s confidence threshold with the teacher’s threshold during each training iteration. We assume that when a student reaches the same confidence level as the teachers in a particular iteration, then there is nothing to be learned for the students from the teachers. This happens in our algorithm 2, when $\tau_s \geq \tau_t$. At this point, adding newer samples to the training set of m_i (the student) would not contribute to its learning anymore. In that sense, we called the point when $\tau_s \geq \tau_t$ as the *graduation point*, so as to stop the tri-training process naturally when the constraint is reached.

3 Evaluation

3.1 Experimental Settings

Datasets. We evaluate our model on the sentiment classification dataset of SemEval-2016 Task 4 Subtask A (Nakov et al., 2016). In total, there are 6000 training sentences, including 3094 positive, 863 neutral, and 2043 negative instances. We use 2000 sentences from the dev set for validation and we have 20632 for test. To test the model’s generalizability, we subsequently examine it under different proportions of labeled data. We select 10%, 20%, 30% and 40% of the training set randomly as labeled samples L and treat the rest as unlabeled U by hiding their labels. Hidden labels are used later for quality check of the generated proxy labels.

Baselines. Since our method improves upon the foundations laid by the typical semi-supervised methods as mentioned in the related work section (e.g. tri-training and self-training), we compare with the following baselines:

1. *NB STr* - Self-training with Naive Bayes as base classifier.
2. *SVM STr* - Self-training with SVM as base classifier.
3. *MLP STr* - Self-training with neural networks (multilayer perceptrons) as base classifier.
4. *Tri* - Tri-training with SVM as base classifiers.
5. *Tri-D* - Tri-training with disagreement with SVM as base classifiers (Søgaard, 2010).

Our proposed approach is tri-training with teacher-student paradigm (Tri-TS). We don’t compare with co-training here because there are no

clear independent views (Zhou and Li, 2005) in the sentiment analysis task. We do not use any deep learning model as base learner in this study, as deep learning models may not perform well in the presence of limited labeled data. We did try FastText (Joulin et al., 2017) as a proof case, but even under the 40% label rate, its performance is unsatisfactory (an initial F_1^{PN} of 0.346 with an improvement of +0.034 using the proposed model).

In all the baselines, we experiment with different base classifiers and their combinations, namely Naive Bayes, SVM and Neural Networks. We use a linear kernel (LinearSVC) for SVM. For the neural networks (MLP), we use 50 neurons in the hidden layer with a softmax output. We use Glove 300-dimensional word embeddings pennington2014glove, . After text-cleaning and tokenization, we average the word-embeddings for the tokens present in the sentence to get the feature vectors. For both the tri-training baselines, Tri and Tri-D, we obtain the best results with SVM as base classifiers. Hence, we report these for comparison with our approach.

Note that, as mentioned in Section 2.3, for the baselines *Tri* and *Tri-D*, we use their own respective stopping criteria during evaluation, as a comparison to our newly proposed stopping criterion.

Parameter Tuning. All parameters required in both the proposed method and the baselines are fine-tuned using the validation set. A grid search is used to determine those parameter values that maximize each model’s performance. For the proposed method τ_t is tuned $\in [0.7, 1.0]$, $\tau_s \in [0.6, 0.95]$. The best performed rates of λ_t and λ_s are found empirically as 0.001. For the tri-training baselines, we try to tune the error constraint as suggested in the original paper, but it generates only small number of proxy labels during the training process and terminates after very limited number of iterations. In that sense, we discard the error constraint and try the threshold based tri-training method as adopted in (Ruder et al., 2017) and (Søgaard, 2010). Best performed parameters are obtained again via evaluations on validation set.

3.2 Results

We evaluate our approach and the baselines from three different aspects: the overall model performance, the quality of generated proxy-labels, and the quantity of unlabeled data consumed. Model performances are reported using F_1^{PN} -score as

	10%	20%	30%	40%
NB STr	0.461	0.471	0.484	0.495
SVM STr	0.465	0.469	0.478	0.489
MLP STr	0.471	0.481	0.497	0.499
Tri	0.478	0.489	0.501	0.505
Tri-D	0.485	0.499	0.507	0.511
Tri-TS	0.498	0.507	0.519	0.523

Table 1: F_1^{PN} comparison averaged over 5 runs for different proportions of labeled data.

adopted in the SemEval competition.

Overall Performance. The methods *Tri* and *Tri-D* both use majority voting to combine the three classifiers. For a fair comparison with these methods, after the training is completed, we perform majority voting on the test set to get the final predictions. In Table 1, we see that the proposed tri-training with teacher-student paradigm consistently outperforms the other baselines with higher prediction performance across different labeled versus unlabeled settings. The proposed method reaches a F_1^{PN} of 0.523 using just 40% of the labeled data, whereas the upper bound F_1^{PN} is only 0.585, if we train the base SVM classifier on the 100% training dataset.

To better understand the effectiveness of the proposed teacher-student paradigm, we further look into the performance of each individual base classifier before the majority voting step. We found that under the 10% label rate, the maximum F_1^{PN} achieved between the base classifiers and the final ensemble model was only 0.011, and such difference decreased to 0.005, when label rate increased to 40%, which indicates good quality of the base classifiers even without the ensemble step. In addition, same conclusion can also be inferred as the base classifiers in Tri-TS before ensemble performed better than the base classifiers in all the other baselines.

Quality of Proxy-labels. The quality of the assigned proxy-labels to the unlabelled data in each iteration determines how well the model learns. So, here, we evaluate the quality of all produced proxy-labels during the self-labeling process against the hidden ground truth to determine the effectiveness of the algorithms in terms of teaching the correct labels. Table 2 shows that teacher models in our proposed method consistently produce high quality proxy-labels (88.93% match with the hided ground truth labels) for the student model to learn. The other baselines tend to suffer from the problem of

	10%	20%	30%	40%
NB STr	65.81	67.14	63.36	70.15
SVM STr	68.15	67.59	71.08	68.13
MLP STr	76.81	77.71	79.07	78.29
Tri	71.78	76.49	75.71	73.35
Tri-D	75.28	70.19	72.37	77.11
Tri-TS	86.18	84.57	88.19	88.93

Table 2: Percentage of matches between the produced proxy-labels and the ground truth averaged over 5 runs for different proportions of labeled data.

adding unreliable labels to the labeled dataset. We view this result as a confirmation of the usefulness of the adaptive threshold in terms of producing high quality proxy-labels on the unlabeled data.

Quantity of Unlabeled Data Consumed. To evaluate the effectiveness of our stopping criterion, we calculate the quantity of unlabeled data consumed during the self-labeling process. Figure 1 shows a plot of the models’ F_1^{PN} with regard to the cumulative number of samples added throughout the iterations (each datapoint in the plot corresponds to an iteration). We find that the proposed method consumes only 201 unlabeled instances to reach the best prediction performance, whereas both the original tri-training and tri-training with disagreement added around twice or thrice the number of samples. From Figure 1, we can further see that many of the baseline algorithms reach the saturation point way before they stop the training process i.e. the improvement in performance is marginal or even decays under some circumstances. This proves the effectiveness of the proposed stopping criteria.

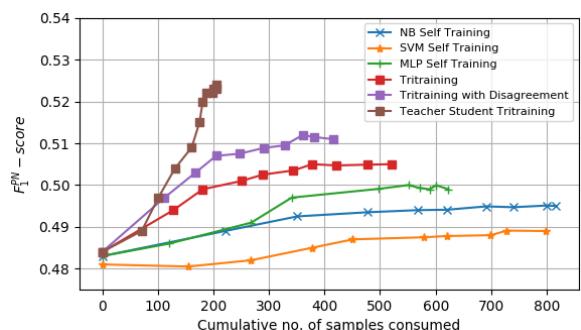


Figure 1: F_1^{PN} score with cumulative number of samples used for all baselines for 40% label rate.

We see that our approach performs worse than the tri-training baselines in the earlier iterations. This happens because our algorithm learns easier

cases in the very beginning and gradually increases the difficulty along the learning process. On the contrary, the original tri-training grows very fast but also plateaus earlier, hence not achieving the full potential of using the three base classifiers. This early plateauing is avoided in our case with the adoption of the adaptive thresholds.

Sensitivity Analysis. We further perform sensitivity analysis for the assessment of the initial settings of τ_t and τ_s with respect to their impact on the model performance. Specifically, we compare the experiment results with: (1) the initial teacher threshold τ_t set over $[0.7, 1.0]$ with initial τ_s fixed as 0.85; and (2) the initial student threshold τ_s set over $[0.6, 0.95]$ with initial τ_t fixed as 0.94. In both settings, τ_t and τ_s are continuously updated with the learned adaptive rates λ_t and λ_s after their initial assignment. We observe only marginal performance losses with an average difference of $-0.015 F_1^{PN}$ over all values. This indicates that the initial value for τ_t and τ_s would not affect the performance that much, as long as they are adaptive.

4 Conclusion

In this paper, we propose a new teacher-student paradigm for original tri-training with continuously adaptive threshold and a natural stopping criteria. We show that our model outperforms all self-training and tri-training baselines in terms of achieving higher overall performance, higher quality of generated proxy labels, while consuming a less quantity of the unlabeled data. Although we only validate the proposed method against the benchmark SemEval dataset in this paper, our ultimate goal is to utilize it as a solution for the scenarios with limited labeled data and to tackle real-world problems, where labeled data is hard to find or expensive to attain.

References

- Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.
- Chien-Lung Chou, Chia-Hui Chang, and Ya-Yun Huang. 2016. Boosted web named entity recognition via tri-training. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(2):10.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April.
- Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)*, pages 1–18.
- Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2017. Knowledge adaptation: Teaching to adapt. *arXiv preprint arXiv:1702.02052*.
- Anders Søgaard. 2010. Simple semi-supervised training of part-of-speech taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 205–208. Association for Computational Linguistics.
- Isaac Triguero, Salvador García, and Francisco Herrera. 2015. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems*, 42(2):245–284.
- Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering*, 17(11):1529–1541.
- Xiaojin Zhu. 2006. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):4.

Generic Web Content Extraction with Open-Source Software

Adrien Barbaresi

Center for Digital Lexicography for the German Language (ZDL)
Berlin-Brandenburg Academy of Sciences (BBAW)
Jägerstraße 22/23 D-10117 Berlin
barbaresi@bbaw.de

Abstract

Web corpus construction involves numerous design decisions. The software packages presented here can help facilitate collection and enhance corpus quality.

1 Problem description

Large “offline” web corpora are now standard throughout disciplines among the research community. Corpus construction notably involves “crawling, downloading, ‘cleaning’ and de-duplicating the data, then linguistically annotating it and loading it into a corpus query tool.” (Kilgarriff, 2007) As such, this process involves a significant number of design decisions and turning points in data processing. Depending on the purpose of data collection, a substantial filtering and quality assessment may also be needed. While some large-scale algorithms can be expected to smooth out irregularities, uses requiring a low margin of error as well as close reading approaches imply constant refinements and improvements in the constitution of the dataset and its processing, for example in the context of an aggregated lexical information platform (Geyken et al., 2017).

Recently, approaches using the CommonCrawl¹ have flourished as they allow for faster download and processing by skipping (or more precisely outsourcing) the crawling phase. Barring the fact that finding one’s “own” way through the Web can be preferable, it is clear that such data should not be used without some filtering. Corresponding to the potential lack of metadata is a lack of information regarding the content, whose adequacy, focus and quality are the object of a post hoc evaluation (Baroni et al., 2009). Because of the vastly increasing variety of corpora, text types and use cases, it becomes more and more difficult to assess the usefulness and appropriateness of certain web texts

for given research objectives. Most notably, an essential operation in corpus construction consists in retaining the desired content while discarding the rest, a polyonymous task referring to peculiar sub-tasks or to the whole, most notably web scraping, boilerplate removal, web page cleaning, or web content extraction (Lejeune and Zhu, 2018).

Consequently, a significant challenge lies in the ability to extract and pre-process web data to meet scientific expectations with respect to corpus quality (Barbaresi, 2019b). In the following, two libraries grounding on previous efforts (Barbaresi, 2016) are presented which can help enhancing the quality of webcorpora. They are both relying on Python, currently one of the most used programming languages, within and outside of academia.²

2 *htmldate*: finding the publishing date

The *htmldate* library (Barbaresi, 2019a) can find both the original and the updated publication dates of web pages. It involves a rule-based examination of the semantic structure of HTML documents, using a combination of tree traversal, common structural patterns, text-based heuristics and robust date extraction. First, it uses the markup in the document header, where common patterns are used to identify relevant elements (e.g. *link* and *meta* elements) including common standards and idiosyncrasies of content management systems. Second, it looks for cues within the HTML code as the whole document is searched for structural markers: *abbr/time* elements and a series of attributes (e.g. *postmetadata*). Finally, a series of heuristics is run on text and markup. The library currently focuses on texts written in English and German, it is used in production and is documented online.³

²Python Software Foundation, <http://www.python.org>
<https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019>

³<https://github.com/adbar/htmldate>

¹<https://commoncrawl.org>

3 *trafilatura*: targeting the main content

The second software component focuses on the main content, which is usually the part displayed centrally, without the left or right bars, the header or the footer, but including potential titles and comments. Distinguishing between whole page and essential parts can help to alleviate many quality problems related to web texts. While this is particularly useful for de-duplication, other tasks related to content extraction also benefit from a cleaner text base. In the concrete case of linguistic and lexicographic research, it allows for content checks on the only portion of the document that really counts.

Although most corresponding Python modules are not actively maintained, the following alternatives perform similar tasks: *dragnet*⁴ features combined and machine-learning approaches, but requires many dependencies as well as extensive tuning; *python-readability*⁵ cleans the page and preserves some markup but is mostly geared towards news texts; *html2text*⁶ converts HTML pages to Markup language and thus keeps the structure, but it doesn't focus on main text extraction. Another issue resides in the lack of output formats corresponding to corpus linguists' needs for document storage and processing, e.g. XML formats such as TEI/XML following the recommendations of the Text Encoding Initiative.⁷

The *trafilatura* library (Barbaresi, 2019c) scrapes the main text of web pages while preserving some structure, which is equivalent to boilerplate removal, DOM-based content extraction, main content identification, and HTML text cleaning. The extraction focuses on original text and can help with the noise consisting of recurring elements (headers and footers, ads, links/blogroll, etc.). It has to be precise enough not to miss texts or discard valid documents, it also has to be reasonably fast, as it is expected to run in production on millions of documents. The processing result can be in plain text or XML format. In the latter case, basic formatting elements are preserved such as text formatting (bold, italic, etc.) and page structure (paragraphs, titles, lists), which can be used for further processing.

This is work in progress⁸, currently experimental

⁴<https://github.com/dragnet-org/dragnet>

⁵<https://github.com/buriy/python-readability>

⁶<https://github.com/Alir3z4/html2text>

⁷<https://tei-c.org>

⁸<https://github.com/adbar/trafilatura>

features include the extraction of comments (separated from the rest), duplicate detection at sentence, paragraph and document level using a least recently used (LRU) cache, TEI/XML output, and language detection on the extracted content.

4 Conclusions

This ongoing work constitutes a step towards the ability to extract and pre-process web texts in order to make them available in clearly definable and coherent collections. In both software components presented here, all the operations needed from web page download to HTML parsing are handled, including scraping and textual analysis. URLs, HTML files or parsed HTML trees are given as input and the libraries output strings in the desired format. They can be used on common operating systems, by themselves, within Python, or on the command-line. Their versatility allows for work on different languages and corpus types as well as for inclusion in various processing chains.

References

- Adrien Barbaresi. 2016. Efficient construction of metadata-enhanced web corpora. In *Proceedings of the 10th Web as Corpus Workshop, Annual meeting of the ACL 2016*, pages 7–16. Association for Computational Linguistics.
- Adrien Barbaresi. 2019a. htmdate. <https://doi.org/10.5281/zenodo.3459599>.
- Adrien Barbaresi. 2019b. The Vast and the Focused: On the need for thematic web and blog corpora. In *Proceedings of the CMLC-7 workshop*, pages 29–32.
- Adrien Barbaresi. 2019c. trafilatura. <https://doi.org/10.5281/zenodo.3460969>.
- Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky Wide Web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.
- Alexander Geyken, Adrien Barbaresi, Jörg Didakowski, Bryan Jurish, Frank Wiegand, and Lothar Lemnitzer. 2017. Die Korpusplattform des "Digitalen Wörterbuchs der deutschen Sprache" (DWDS). *Zeitschrift für germanistische Linguistik*, 45(2):327–344.
- Adam Kilgarriff. 2007. Googleology is bad science. *Computational Linguistics*, 33(1):147–151.
- Gaël Lejeune and Lichao Zhu. 2018. A New Proposal for Evaluating Web Page Cleaning Tools. *Computación y Sistemas*, 22(4).

Ein Tool zur Visualisierung des Gebrauchs von Schreibvarianten

Peter M. Fischer

Leibniz-Institut für Deutsche Sprache

R 5 6-13

68161 Mannheim

peter.fischer@ids-mannheim.de

Christian Lang

Leibniz-Institut für Deutsche Sprache

Augustaanlage 32

68165 Mannheim

lang@ids-mannheim.de

Abstract

In unserem Beitrag stellen wir die Entwicklung eines komponentenbasierten Tools zur Abfrage, Auswertung und Visualisierung von Schreibvarianten vor.

1 Einleitung

Die diachrone empirische Untersuchung von Varianz in der Schreibung von Wörtern anhand von Korpusdaten spielt eine zentrale Rolle in der Orthografieforschung und wird auch in Untersuchungen der AG Korpus des Rats für deutsche Rechtschreibung angewendet (vgl. Krome/Roll 2016: 7). Dazu werden in Einzelrecherchen aussagekräftige Grafiken erstellt, die für gegebene Schreibvarianten unter gegebenen Schreibern und in einem gegebenen Untersuchungszeitraum eine übersichtliche Gegenüberstellung entsprechender Schreibgebräuche ermöglichen (vgl. Abb.1).

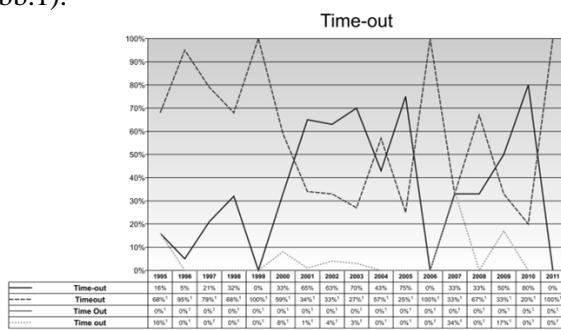


Abb. 1: Erhebung der Vorkommen der Schreibungen „Time-out“, „Timeout“, „Time Out“ und „Time out“ (Rat für deutsche Rechtschreibung)

Der Erstellung solcher Grafiken geht ein mehrstufiger Prozess voraus (Auswahl der Korpusgrundlage, Erstellung und Durchführung von Korpusabfragen, Zusammentragen der Abfrageergebnisse, Erstellung der Grafiken), der an verschiedenen Stellen nicht oder nur teilweise automatisiert ist. In Konsequenz wird dieser Pro-

zess daher stark indikations- und bedarfsgeleitet angestoßen, was ein flächendeckendes Monitoring des allgemeinen Schreibgebrauchs ausbremsst. Um den Rat für deutsche Rechtschreibung in seiner Arbeit zu unterstützen, wird am Leibniz-Institut für Deutsche Sprache (IDS) eine Software entwickelt, die diesen Prozess durch breitere Automation in den Abläufen fördert.

2 Ziele und Anforderungen

Eine Kernfunktionalität des Tools liegt in der interaktiven Erstellung von Schaubildern, welche die Entwicklung benutzerdefinierbarer Häufigkeitsmaße von Schreibvarianten entlang einer Zeitachse abbilden. Die Benutzeroberfläche begleitet Anwender/-innen dabei im gesamten Erstellungsprozess von der Korpusauswahl bis zum Grafikexport und bietet einen intuitiven und autonomen Zugang zu Untersuchungsergebnissen.

Im Hinblick auf die statistischen Auswertungen und Diagrammtypen greift das hier vorgestellte Tool die Methodologie der auch am IDS entwickelten Programm-Infrastruktur¹ zur Generierung sog. Zeitverlaufsgrafiken (ZVGs) auf, knüpft aber an die ebenfalls am IDS entwickelte universelle Korpusanalyseplattform *KorAP* (Kupietz et al. 2019) an. Da *KorAP* als quelloffenes Projekt entwickelt wird, besteht keine Abhängigkeit zu IDS-Infrastrukturen. Zudem kann *KorAP* auch auf physisch verteilt liegenden Ressourcen operieren, sodass auch der Standort der auszuwertenden Korpora nicht an das IDS gebunden ist, was den Bedürfnissen des Rats für deutsche Rechtschreibung insbesondere in Bezug auf Untersuchungen in bzw. aus den Gebieten seiner sieben internationalen Mitglieder gerecht wird.

Die **einfach** strukturierte und leicht zugängliche Benutzeroberfläche befähigt Nutzer/-innen, auch ohne größere Kenntnis von Korpusrecherche, Datenauswertung und -visualisierung komfortabel vergleichende Grafiken zu erstellen.

¹ <http://www1.ids-mannheim.de/kl/projekte/methoden/mdca/zvgs.html>

Ein- und Ausgabedaten können **flexibel** festgelegt und an das jeweilige Auswertungsszenario angepasst werden. Aufseiten der Eingabe sind die Auswahl zur Verfügung stehender Korpora und ihre weitere Eingrenzung zu bestimmen (siehe virtuelle Korpora, Diewald et al. 2016) und eine Liste zu untersuchender Schreibvarianten anzugeben. Aufseiten der Ausgabe können die Art der Darstellung in Form unterschiedlicher statistischer Maße wie absolute Frequenz, relative Frequenz, Häufigkeitsklassen (vgl. Perkuhn et al. 2012:80), prozentuale Verteilung der Varianten etc. sowie beim Export der erstellten Visualisierung das Dateiformat (u.a. png, jpg, pdf, svg, xml oder html) gewählt werden.

Die Visualisierung wird bei anwenderseitig nachträglich eingebrachten Änderungen der Darstellungskriterien **dynamisch** angepasst. Dies erlaubt eine rasche und intuitive Justierung der Grafik bis zum gewünschten Bild.

3 Software-Architektur

Das Tool sieht sich als stark spezialisierte Abfrage-, Auswertungs- und Darbietungsplattform und knüpft daher auf der einen Seite an bestehende Schnittstellen zur allgemeinen Korpusabfrage an und bildet auf der anderen Seite selbst eine Schnittstelle zur Benutzerinteraktion.

Zur Erlangung der **Datengrundlage** bedient sich das Tool skriptgesteuert einer Programmierschnittstelle (*API*) der Korpusanalyseplattform *KorAP* (Kupietz et al. 2019), die für die Verarbeitung sehr großer Korpora mit mehreren Annotationsebenen, mehreren Abfragesprachen und komplexen Lizenzmodellen optimiert ist und in Modulen als Open Source auf GitHub² veröffentlicht wird. Über die am IDS installierte Instanz³ besteht somit u.a. Zugang zum Deutschen Referenzkorpus *DeReKo* (Kupietz et al. 2018).

Die **Visualisierung** ist als Webapplikation in R (R Core Team 2016) mithilfe des R-Paketes *shiny* (Chang et al. 2019) implementiert, das eine interaktive Datenpräsentation in einer Weboberfläche und damit dynamische Anpassungen gemäß Anwendereingaben ermöglicht.

4 Ausblick

Perspektivisch ist geplant, das Tool auch als Modul von *grammis*⁴, dem grammatischen Informationssystem des IDS, der Öffentlichkeit

zugänglich zu machen. So soll es die Anwendung *KoGra-R* (Hansen-Morath et al. 2019)⁵ komplementieren, die ebenfalls am IDS entwickelt wurde und korpuslinguistische Statistiken und Visualisierungen bereithält, allerdings nicht für den Variantenvergleich konzipiert wurde.

Literaturangaben

- Chang, W./Cheng, J./Allaire, JJ/Xie, Y./McPherson, J. 2019. *shiny: Web Application Framework for R*. <https://cran.r-project.org/web/packages/shiny/shiny.pdf>, R package version 1.3.2.
- Diewald, N./Hanl, M./Margaretha, E./Bingel, J./Kupietz, M./Bański, P./Witt, A. 2016. KorAP Architecture – Diving in the Deep Sea of Corpus Data. In: Calzolari, N. et al. (Hrsg.). 2016. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Portorož, Slovenia*. Paris: European Language Resources Association (ELRA), S. 3586–3591.
- Hansen-Morath, S./Schmitz, H-C./Schneider, R./Wolfer, S. 2019. KoGra-R: Standardisierte statistische Auswertung von Korpusrecherchen. In: Fuß, E./Konopka, M./Wöllstein, A. (Hrsg.). 2019. *Grammatik im Korpus*. Tübingen: Narr. S. 299–357.
- Krome, S./Roll, B. 2016. Fremdwörter zwischen Isolation und Integration. Empirische Analysen zum Schreibbusus auf der Basis von Textkorpora professioneller und informeller Schreiber. In: *Studia Germanistica 19/2016*. Ostrava: Ostravská univerzita. S. 5–40.
- Kupietz, M./Lüngen, H./Kamocki, P./Witt, A. 2018. The German Reference Corpus DeReKo: New Developments – New Opportunities. In: Calzolari, N. et al. (Hrsg.). *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki: European Language Resources Association (ELRA), S. 4353–4360.
- Kupietz, M./Diewald, N./Margaretha, E./Bodmer, F./Stallkamp, H./Harders, P. 2019. Neues von KorAP. In: Eichinger, L. M./Plewnia, A. (Hrsg.): *Neues vom heutigen Deutsch. Empirisch – methodisch – theoretisch. Jahrbuch des Instituts für Deutsche Sprache 2018*. Berlin/Boston: de Gruyter. S. 345–349.
- Perkuhn, R./Keibel, H./Kupietz, M. 2012. *Korpuslinguistik*. Paderborn: Fink, 2012.
- R Core Team. 2016. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, URL <https://www.R-project.org/>.
- 5 <http://kograno.ids-mannheim.de/index.html>

² <http://github.com/KorAP>

³ <https://korap.ids-mannheim.de/>

⁴ <https://grammis.ids-mannheim.de>

Identification of Reading Absorption in User-Generated Book Reviews

Piroska Lendvai, Simone Rebora, Moniek Kuijpers

Digital Humanities Lab
University of Basel, Switzerland
piroska.r@gmail.com

Abstract

We introduce a language processing approach to detect reading absorption expressed in book review texts in English from an online social reading platform. Such texts are opinionated, subjective, variable length self-narratives. We describe our corpus annotated with absorption categories that were defined in empirical aesthetics, based on which we performed supervised, sentence level, binary classification of the presence or absence of reading absorption, using text-based, distributional features.

1 Introduction

Our study aims to contribute to affect recognition research by introducing the affective state of absorption during reading fiction. Our goal is to computationally process user-generated book reviews from an online social reading community, to identify passages that textually express reading absorption. The reviews belong to the genre of self-narratives that report about individual experiences in a non-elicited way, typically serving multiple user intents such as providing one or more of evaluation, recommendation, feedback, as well as influencing and socializing. The reviews often do not merely contain mentions of evaluative sentiment toward (components of) the book, but rather also express complementary aspects in terms of engagement of the reader, for example immersive experiences (*"I was glued to my kindle."*; *"It stayed with me, even when I wasn't reading it"*), transportation to the fictional world (*"i felt like am living inside it"*), altered sense of time during reading (*"it is almost 800 pages that just fly by"*), emotional engagement (*"I cried reading the last 30 pages"*), and others.

Computational analysis of detecting reader absorption has so far been largely unaddressed, except

for our initial text similarity approach to detect absorption in the story world (Rebora et al., 2018). In this study, we are interested in Natural Language Processing (NLP) based modeling of the specific affective state of reading absorption. Our corpus construction is currently ongoing: the experiments were based on 200 reviews, from which we generated the first processing resources, i.e., distributional language models and supervised classifiers, to benefit researchers in computational linguistics, literature and social sciences studies.

2 Corpus and Labeling

Our current corpus consists of 200 English review texts which we collected from a social reading platform. These reviews pertained to books from different literary genres (romance, fantasy, thriller) that we pre-selected based on high star-ratings on the platform and the presence of trigger words. The data were balanced for amount of review per book.

We trained five annotators for labeling absorption in terms of a taxonomy of roughly 40 fine-grained absorption labels, grouped under broad concepts such as Attention, Transportation, Emotional Engagement, Mental Imagery, Disconnection from reality, etc. taken from Kuijpers et al. (2014) and Bálint et al. (2016). The annotators could also mark up when users explicitly signaled the lack of absorption (e.g. *"I struggled to get through a lot of the pages"* or *"None of the characters really mattered to me"*), to make them distinct from expressions of the presence of absorption.

The annotators worked on the review level using Brat¹ and could assign labels to text segments of arbitrary length. For the current study, we aggregated all annotators' labels into a generic absorption category: the *Abs* label was assigned if at least one of five annotators judged some part of a sentence as explicitly expressing some type of absorption

¹<https://brat.nlplab.org/>

Sent	Text	Fine-grained label (Support)	Binary label	Justification
1	FUNNY.	-	nonAbs	generic evaluation
2	Funny funny funny and sexy as hell.	-	nonAbs	generic evaluation
3	I don't only like the Heroine,	-	nonAbs	generic sentiment
4	I LOVE her.	-	nonAbs	generic sentiment
...
12	<i>I want to be Molly when I grow up.</i>	Wishful identification (4)	Abs	Wish of having the same characteristics as protagonist
13	I loved her backstory and why she is the way she is.	-	nonAbs	generic sentiment/evaluation
14	Her Career Secret frustrated me at times.	-	nonAbs	generic sentiment
15	<i>Most of the time, I was like, "JUST TELL HIM."</i>	Participatory response (1)	Abs	Intention to intervene, at times by addressing the characters directly
16	<i>But I got why she felt she couldn't.</i>	Emotional understanding (2)	Abs	Emotional or cognitive understanding of the character's feelings or perspective
17	This is a great book.	-	nonAbs	generic evaluation

Table 1: Corpus excerpt with fine-grained absorption annotations (column 2) and the binarized target labels to classify (column 3). *Abs*: reading absorption or its lack is expressed, *nonAbs*: no absorption or its lack is expressed.

or the lack of it. Sentence-level segmentation was obtained using the Spacy package² that worked best for our user generated text type. In the pilot annotation round, the average review length was 25 sentences (stdev ± 28), inter-annotator agreement on the sentence level was 0.59 (Fleiss' Kappa).

To illustrate our data and classification task, a review excerpt is presented in Table 1, in which e.g. sentence "*I want to be Molly when I grow up.*" was judged as *Wishful identification* by four annotators.

3 Reading Absorption Identification

The current dataset is imbalanced, as only 13% of the instances have the target class *Abs* (660 vs 4,327 sentences). We used a random undersampling method³ during training to account for it. Sentences were stripped of punctuation, tokens were lowercased and stemmed (mean normalized sentence length: 15 ± 13 tokens), and represented in terms of a count vector (length: 6,064) as well as a sentence embedding vector (length: 100). We generated the sentence embedding representation using the *sent2vec* tool⁴ that we retrained on 2.45 million unlabeled social reading narratives collected from the online platform.

Next, we performed classification experiments using two classical machine learners with no optimization: logistic regression (class_weight=balanced) and random forest, in 5-fold cross-validation. The feature sets representing the sentences were tested in isolation and in combination. The results are presented in Table 2 and show that good precision is difficult to achieve for the target class in the current setup: the best F-

scores are .42 using the large bag of words count vector or using all features. We are currently growing the corpus and consolidating the still evolving labeling scheme, after which we will be able to test more advanced data representation and learning approaches, and evaluate classification on the fine-grained absorption labels.

LR	Abs	nonAbs	RF	Abs	nonAbs
cv	P: 0.30 R: 0.70 F: 0.42	0.94 0.75 0.84	cv	P: 0.30 R: 0.62 F: 0.40	0.93 0.78 0.85
s2v	P: 0.24 R: 0.73 F: 0.37	0.94 0.65 0.77	s2v	P: 0.26 R: 0.81 F: 0.39	0.96 0.64 0.77
all	P: 0.29 R: 0.76 F: 0.42	0.95 0.72 0.82	all	P: 0.25 R: 0.79 F: 0.38	0.95 0.65 0.78

Table 2: Classification results by Logistic Regression (LR) and Random Forest (RF) in terms of Precision, Recall, and F-score per class, averaged from 5-fold cross-validation. Features: CountVectorizer (cv) and sent2vec embeddings (s2v).

References

- Bálint, K., Hakemulder, F., Kuijpers, M., Doicaru, M., Tan, E. S. (2016). Reconceptualizing foregrounding. *Scientific Study of Literature*, 6(2), 176-207.
- Kuijpers, M., Hakemulder, F., Tan, E.E. and Doicaru, M.M. (2014). Exploring absorbing reading experiences. Developing and validating a self-report scale to measure story world absorption. *Scientific Study of Literature*, 4(1): 89122.
- Rebora, S., Lendvai, P. and Kuijpers M. (2018). Reader experience labeling automatized: Text similarity classification of user-generated book reviews. In: EADH 2018 Conference, Book of Abstracts.

²<https://spacy.io>

³<https://github.com/scikit-learn-contrib/imbalanced-learn>

⁴<https://github.com/epfml/sent2vec>

Sketches of a graphical user interface for word alignment annotation

Maria Skeppstedt, Magnus Ahltorp, Gunnar Eriksson, Rickard Domeij

The Language Council of Sweden, The Institute for Language and Folklore, Sweden

firstname.lastname@sprakochfolkminnen.se

Abstract

We describe ideas for the graphical user interface of a tool for word alignment annotation. Our prototype interface builds on a design template from the Jigsaw system for investigative analysis, and is implemented as a web application.

1 Introduction

Although many models for machine translation no longer rely on training data in the form of word-aligned corpora, the concept of word alignment – and thereby the need for manually aligned corpora – still exists. For instance, manually annotated corpora are required for evaluating the alignment quality of translation systems that use word alignment as an intermediate step (Alkhouri et al., 2016) or systems that use automatically aligned corpora for dictionary construction (Bourgonje et al., 2018).

There exist several tools for word alignment annotation. Many of these tools are, however, either (i) several years old, and to the best of our knowledge no longer updated (Tiedemann, 2002; Zhang et al., 2008) or, (ii) targeting a more complex annotation task, in which the word alignment task is only a subtask (Hung-Ngo and Winiwarter, 2012; Wirén et al., 2018).

We therefore aim to construct an annotation tool that uses current libraries¹ for web development, and which is solely focused on the task of word alignment in sentence-aligned texts.

2 Design ideas for the user interface

Our prototype interface for word alignment annotation builds on a design template from the Jigsaw system for investigative analysis. This template has previously been used for visualising associations between entities extracted from text collections (Stasko, 2008; Skeppstedt et al., 2018). The

template dictates that the entities, which here correspond to words that are to be aligned, are displayed in separate lists, and that associations between elements in the different lists are indicated by lines that connect them and by highlighting.

By arranging the words vertically, the display of the word associations becomes more compact for most writing systems, which we hypothesise will make it easier to trace the connecting lines. While this potentially de-emphasises the sentence, it instead emphasises the individual tokens.

Figure 1 shows the prototype applied to one sentence-pair in a corpus of parallel texts collected from translations made at Swedish government agencies (Dahlberg and Domeij, 2017).

The prototype interface contains the following components: (a) The sentence in the first language. (b) The sentence in the second language. (c) An alignment between two words is created by drag-and-drop, i.e. by dragging an element in the left-hand list and dropping it on an element in the right-hand list. (d) Alignment is shown by a line that connects the two list elements. (e) In addition, when the user hovers the mouse over an element in one of the lists, the associated elements in the other list are highlighted. (f) An alignment is removed by clicking on the corresponding delete button. (g) Drop-down list for choosing which corpus to annotate. (h) Drop-down list for choosing the criterium² by which the next sentence-pair to annotate is to be selected. (i) The user can choose either annotation mode or to browse previously annotated sentences in read-only mode. (j) Swap the two languages. (k) Save annotation. (l) Go back to the previously annotated sentence-pair. (m) Remove sentence-pairs from the annotation task (e.g. when the sentence-pair stems from an incorrect sentence alignment).

²We plan to add the functionality of pre-annotated alignments. The user should then be able to select the order in which the sentence-pairs are to be annotated, e.g. to choose to start with the ones that the pre-alignment system estimates to be easiest or estimates to be most difficult.

¹D3 and Flask.

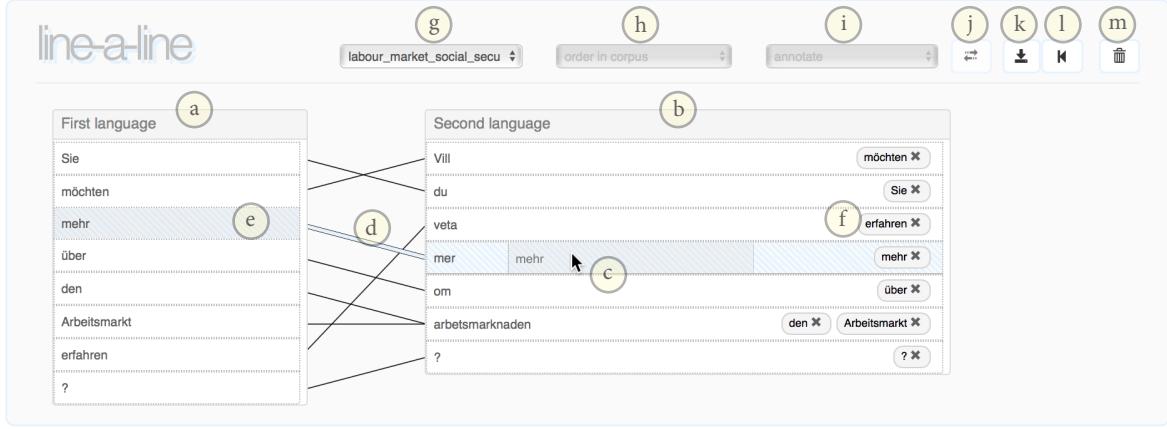


Figure 1: User interface prototype, showing a Swedish-German sentence-pair annotated for word alignment.

3 Next steps

The immediate next steps will consist of refining and evaluating the user interface. We also plan to evaluate the usefulness of the alignment functionality for tasks beyond core natural language processing, e.g. usefulness for translation studies and for research on the application of official terminologies in translated texts (Dahlberg, 2017).

The annotation tool should also be able to provide an optional automatic pre-alignment, i.e. an alignment on which the user can base their manual annotation. To enable pre-alignment for language pairs for which no large parallel corpora exist, the alignment functionality should preferably be based on methods suitable also for small corpora.

Acknowledgements

This work was funded by the Swedish Research Council (project number 2017-00626).

References

- [Alkhouri et al.2016] Tamer Alkhouri, Gabriel Bretschner, Jan-Thorsten Peter, Mohammed Hethnawi, Andreas Guta, and Hermann Ney. 2016. Alignment-based neural machine translation. In *ACL 2016 First Conference on Machine Translation*, pages 54–65.
- [Bourgonje et al.2018] Peter Bourgonje, Jet Hoek, Jacqueline Evers-Vermeul, Gisela Redeker, Ted Sanders, and Manfred Stede. 2018. Constructing a lexicon of dutch discourse connectives. *Computational Linguistics in the Netherlands Journal*, 8:163–175, 12/2018.
- [Dahlberg and Domeij2017] Simon Dahlberg and Rickard Domeij. 2017. Översättning av termer i myndighetstexter: En studie om översättning av myndighetstermer i arbetet med nationell språkinfrastruktur på språkrådet. In *Workshop Termplanering och termbruk i svenska på Svenskans beskrivning 36*.
- [Dahlberg2017] Simon Dahlberg. 2017. Tre svenska myndigheters strategier för termöversättning till spanska och franska. Master’s thesis, Stockholm University, Department of Linguistics.
- [Hung-Ngo and Winiwarter2012] Quoc Hung-Ngo and Werner Winiwarter. 2012. A visualizing annotation tool for semi-automatically building a bilingual corpus. In *The Fifth Workshop on Building and Using Comparable Corpora*, pages 67–74.
- [Skeppstedt et al.2018] Maria Skeppstedt, Kostiantyn Kucher, Manfred Stede, and Andreas Kerren. 2018. Topics2Themes: Computer-Assisted Argument Extraction by Visual Analysis of Important Topics. In *Proceedings of the LREC Workshop on Visualization as Added Value in the Development, Use and Evaluation of Language Resources*, pages 9–16.
- [Stasko2008] John Stasko. 2008. Jigsaw: Investigative analysis on text document collections through visualization. In *DESI II: Second International Workshop on Supporting Search and Sensemaking for Electronically Stored Information in Discovery Proceedings*.
- [Tiedemann2002] Jörg Tiedemann. 2002. Uplug a modular corpus tool for parallel corpora. *Language and Computers*, pages 181–197, 07.
- [Wirén et al.2018] Mats Wirén, Arild Matsson, Dan Rosén, and Elena Volodina. 2018. Svala: Annotation of second-language learner text based on mostly automatic alignment of parallel corpora. In *Selected papers from the CLARIN Annual Conference 2018*, number 159, pages 227–239. Linköping University Electronic Press, Linköpings universitet.
- [Zhang et al.2008] Yujie Zhang, Zhulong Wang, Kiyotaka Uchimoto, Qing Ma, and Hitoshi Isahara. 2008. Word alignment annotation in a Japanese-Chinese parallel corpus. In *LREC 2008*.

Predicting default and non-default aspectual coding: Impact und Density of information features¹

Michael Richter Tariq Yousef

Universität Leipzig

Natural Language Processing Group

{richter, tariq}@informatik.uni-leipzig.de

Abstract

This paper presents a study on the automatic classification of default and non-default codings for aspect-marked verbs in six Slavic languages and in Latvian. As classifier a Support Vector Machine and as verbal features *Shannon Information* (SI) and *Average Information Content* (IC) have been utilised. In all languages high accuracy of the classification has been achieved. In addition, we found indications for the validity of the *Uniform Information Density principle* within SI and IC.

1 Introduction

The research questions are: can Shannon's theorem be transferred to natural languages, and, in particular, does coding of aspect marked verbs interact with the information that they carry?

Our point of departure is that verbs have a dominant aspect category and that this category can be determined by frequency distributions: default forms will occur more frequently than non-default forms.

2 Aims, Data and Method

The first aim of the study is to test whether default and non-default coding of aspect-marked verbs in the six Slavic languages Bulgarian, Old Church Slavonic, Polish, Slovak, Slovenian, Ukrainian and Latvian can be classified by two verbal information features: *Average Information Content* (henceforth 'IC') (Cohen Priva, 2008; Piantadosi et al., 2011, see (1))

$$IC = E(-\log_2(P(W = w | context))) \quad (1)$$

As contexts, we took bigrams (*lexical surprisal*, Hale, 2001; Levy, 2008; Levy, 2013), to both directions of the target verbs as a study of Richter et al. (2019) disclosed that target verbs convey the highest amount of information in bigram contexts. We took *Shannon Information* (henceforth 'SI', Shannon and Weaver, 1948) as the negative log probability of a target verb form in the corpus.

The aim and the choice of the two information-theory based features are motivated by Shannon's *source coding theorem* (Shannon and Weaver, 1948) on the interaction of information, coding and length of signs. As classifier we employed a Support Vector Machine (SVM) binary classifier with a radial basis function kernel (Joachims, 1998).

The second aim of the study is to test whether the *Uniform Information Density* – hypothesis, (henceforth UIDh; Genzel and Charniak, 2002; Aylett and Turk, 2004; Levy and Jaeger, 2007; Jaeger, 2010), holds within the features IC and SI of the target verbs. In its original form, UIDh is applied to discrete signs: there should neither be extreme peaks nor extreme troughs in the stream of information in order to facilitate language processing. We, however, apply UIDh to two different information values of a *single* sign and hypothesise based on previous research (Celano et al., 2018) that the variances in information density within IC and SI should tend towards zero (Collins, 2014). We utilised *Global Information Density* UID_{GLOBAL} (see (3)): id_i is the information density of SI and IC of a single verb form, and μ is the mean of id :

$$UID_{GLOBAL} = -E(\sum_{i=1}^N id_i - \mu)^2 \quad (3)$$

As data resource we exploited Universal Dependency Treebanks (version 2.3, <https://universaldependencies.org>) because verbal aspect is encoded

¹ Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) project number: 357550571.

in these corpora. For each verb, the default and non-default aspect was determined. The number of tokens and the numbers of word forms, respectively, for each language are: Bulgarian 156,149 / 13,714, Old Church Slavonic 57,563 / 9,575, Polish 1498042 / 7,199, Slovak 10,6043 / 11,749, Slovenian 170,158 / 11,629, Ukrainian 122,275 / 9,789 and Latvian 208,965 / 17,046. We reduced aspect oppositions to the binary imperfective-perfective distinction, and took the difference of both occurrences. The differences were normalized, and ten thresholds between [.09:1] were set.

3 Results

We focused on the thresholds in the interval [.19, .59] in order to ensure a sufficient number of default and non-default encodings for the training of the SVM-classifier: the accuracy is almost independent of the threshold and thus of the frequency distribution: even with an almost equal distribution of default and nondefault aspect frequencies that is, with threshold .19, almost perfect accuracy values are achieved. The range of accuracy in [.19,.59] is: Bulgarian 99.5 – 99.8, Old Church Slavonic 94.3 – 97.8, Polish 99.7 – 99.9, Slovak 99.5 – 99.6, Slovenian 100 – 100, Ukrainian 99.1 – 100 and Latvian 98.3 – 99.5. Estimating UID-GLOBAL to our test set of languages, an identical pattern in all languages comes to light: the majority of variance values tends to be close to zero.

4 Conclusion

As Shannon's source coding theorem predicts, we found interactions of aspectual coding and information: Our study provides evidence that non-default coded verb forms are more informative than default forms. Almost identical accuracy has been achieved with all tested threshold values.

With regard to the second aim, our study discloses that UIDh holds within IC and SI: both features convey a uniform stream of information throughout the verb forms of the seven languages in focus.

The practical impact of our study concerns the assignment of word classes in languages such as Tagalog: default and non-default forms of a lemma correspond with different word classes.

References

Matthew Aylett and Alice Turk. 2004. The Smooth Signal Redundancy Hypothesis: A functional

- explanation for relationships between redundancy, prosodic prominence, and duration in spontaneous
- Guiseppe Celano, Michael Richter, Rebecca Voll, and Gerhard Heyer. 2018. Aspect coding asymmetries of verbs: The case of Russian. *KONVENS 2018. PROCEEDINGS of the 14th Conference on Natural Language Processing*, 34 – 39.
- Uriel Cohen Priva. 2008. Using information content to predict phone deletion. *Proceedings of the 27th West Coast Conference on Formal Linguistics*: 90 –98..
- Michael Xavier Collins. 2014. Information density and dependency length as complementary cognitive models. *Journal of Psycholinguistic Research*, 43(5): 651 – 681.
- Genzel, Dmitriy and Eugene Charniak. 2002. Entropy rate constancy in text. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, July 2002, pages 199 – 206.
- John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. *Proceedings of NAACL*: 1 – 8.
- T. Florian Jaeger. 2010. Redundancy and reduction: Speakers manage syntactic information density. *Cognitive Psychology*, 61 (1): 23 – 62. doi: 10.1016/j.cogpsych.2010.02.002.
- Thorsten, Joachims. 1998. *Text categorization with Support Vector Machines: Learning with many relevant features* Retrieved from http://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf.
- Roger Levy and T. Florian Jaeger. 2007. Speakers optimize information density through syntactic reduction. In *Proceedings of the 20th Conference on Neural Information Processing Systems (NIPS)*.
- Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106: 1126–1177.
- Roger Levy. 2013. Memory and Surprisal in Human Sentence Comprehension. In Roger van Gompel, (ed.), *Sentence Processing*: 78 – 114. Psychology Press, Hove.
- Steven T. Piantadosi, Harry Tily and Edward Gibson. 2011. Word lengths are optimized for efficient communication. *PNAS*, 108(9): 3526 –3529.
- Michael Richter, Yuki Kyogoku and Max Kölbl. 2019. Interaction of Information Content and Frequency as predictors of verbs' lengths. In Witold Abramowicz and Rafael Corchuelo (eds.), *Business Information Systems*. Springer, 271 – 282.
- Claude E. Shannon, and Warren Weaver. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27.

A New German Reddit Corpus

Andreas Blombach and **Natalie Dykes** and **Stefan Evert** and
Philipp Heinrich and **Besim Kabashi** and **Thomas Proisl**

Lehrstuhl für Korpus- und Computerlinguistik
Friedrich-Alexander-Universität Erlangen-Nürnberg
Erlangen, Germany

{andreas.blombach, natalie.mary.dykes, stefan.evert}@fau.de
{philipp.heinrich, besim.kabashi, thomas.proisl}@fau.de

Abstract

We describe the creation of a German Reddit corpus, difficulties encountered along the way, and some of the data’s linguistic peculiarities.

1 Data Gathering

What’s Reddit? Reddit is a platform combining social news aggregation, discussion and microblogging. Since its founding in 2005, it has grown to be one of the most popular websites in the USA; in recent years, its popularity has also increased in Germany, as indicated by site rankings from Alexa and SimilarWeb.¹

Users submit content (e.g. text, images or links) and can comment on submissions. Submissions and comments can be voted up or down, affecting the order in which they are displayed (submissions with the most upvotes make it to the front page).

Reddit is structured into so-called “subreddits” with their own community rules. Subreddits range from being rather open-topic (e.g. *r/de* – anything related to German) to extremely specific.

The German Reddit Sphere Just as subreddits’ topics and contents vary widely, so do linguistic phenomena associated with particular subreddits. While some subreddits exhibit mostly standard language, others have rather unique memes and practices; making them difficult for outsiders to understand. In German subreddits, for instance, emoticons may be replaced by German *Umlaut* characters:

:o → Ü :o → Ö :< → Ä

On a lexical and phraseological level, typical expressions commonly associated with online communication as well as Reddit-specific ones are often

¹Ranks as of October 9, 2019:
Alexa: 6 (US), 9 (DE); SimilarWeb: 10 (US), 31 (DE); see <https://www.alexa.com/topsites/countries> and <https://www.similarweb.com/top-websites>

translated word for word, leading to a humorous effect: *pfstieren* ‘to post’, *ausgelöst* ‘triggered’, *Unterlases* ‘subreddit’, *fixierte das für dich* ‘fixed this for you’.

2 Corpus Creation

Building a Reddit Corpus In an ongoing effort, Jason Baumgartner collects every Reddit submission and comment, publicly accessible via <https://files.pushshift.io/reddit/>² (some caveats apply, see Gaffney and Matias (2018)). The first attempt at extracting German comments was made by Barbaresi (2015). At the time, Reddit was less widely used, especially by German-speaking users, and the resulting corpus was relatively small (97,505 comments, 566,362 tokens).

Detecting German Comments To detect German comments in the vast dataset, we adapt Barbaresi’s approach of using a two-tiered filter relying on spell checking and language identification. After some sanitizing steps to ignore extremely short or deleted comments, every token is checked against a German and an English dictionary with a regular expression. A text is classified as potentially German if at least 70% of its tokens are found in the dictionary, and no more than 30% are present in the English dictionary. Next, *langid* (Lui and Baldwin, 2014) is run on these candidate comments to ultimately classify comments as German. This way, we identified more than 6,700,000 German comments between 2016 and 2018, amounting to roughly 230,000,000 tokens running text.

Evaluation of the Two-tiered Filter In a random sample of 1,618 comments, we manually identified 26% which are not actually in German. While longer comments are recognized rather reliably,

²See also https://www.reddit.com/r/pushshift/comments/bcxguf/new_to_pushshift_read_this_faq/

Subreddit	“German”	Total	Fraction
de	3,644,481	4,364,440	0.835
Austria	389,041	483,125	0.805
rocketbeans	142,787	164,699	0.867
AskReddit	125,411	183,147,454	0.000685
edefreiheit	121,370	147,150	0.825

Table 1: Comment counts in the top 5 subreddits with “German” comments, 2016–2018

shorter comments are often misclassified – for instance, due to the presence of proper names.

Possible Corpus Cleanup Table 1 shows the top 5 subreddits containing comments identified as German. We interpret the low relative frequencies of German comments in some subreddits as an indication of false positives. Thus, filtering out subreddits with less than e.g. 10% of German comments seems like a plausible strategy. On the other hand, for many predominantly German subreddits, the two-tiered filter may have been too aggressive, and it might be sensible to retain all comments from these to keep conversations intact.³ In any case, using the filter to identify predominantly German subreddits in the first place works as intended.

3 Annotation

Pre-processing and Tokenization Some of Reddit’s peculiarities pose challenges to existing tools. Comments can include Markdown markup, making pre-processing necessary. Short form links to subreddits and user profiles follow the pattern *r/oldschoolcool* and *u/username*, which has to be accounted for by a tokenizer. Punctuation is often omitted and replaced by line breaks or emoticons: *So bin endlich zu was gekommen :D Habe [...]*

Tokenization and POS Tagging A random sample of comments was tokenized using SoMaJo (Proisl and Uhrig, 2016) and tagged with the STTS_IBK tagset (Beißwenger et al., 2015) using SoMeWeTa (Proisl, 2018). To evaluate performance, manual correction was performed on 1,186 tokens. The tagging accuracy was at 92.6%. POS errors seemed to be largely systematic, with the very fine-grained differentiation in particle types being hard to achieve due to sparseness in the training data, i.e. the EmpiriST corpus (Beißwenger et al., 2016).

Our evaluation leads us to the question whether a revised CMC tagset might be beneficial: While

³Subreddits containing dialectal language use (such as *r/aeiou* or *r/BUENZLI*) are a special case.

there are fine-grained categories for e.g. particle types, more obvious distinctions are not made (e.g. between definite and indefinite articles). Moreover, only certain contractions are assigned tags, and no differentiation is made for common acronyms (*scnr; imho*) and different types of punctuation (also affecting asterisks marking “action words” like **lol**).

4 Outlook

Due to their peculiarities, Reddit data are a promising source for further (socio-)linguistic research. Since the same features pose challenges to existing tools, more corpus cleaning will be necessary and rules for tokenization and tagging will need to be updated.

References

- Adrien Barbaresi. 2015. Collection, description, and visualization of the German Reddit corpus. In *Proceedings of the 2nd Workshop on Natural Language Processing for Computer-Mediated Communication / Social Media*, pages 7–11, Essen.
- Michael Beißwenger, Thomas Bartz, Angelika Storrer, and Swantje Westpfahl. 2015. Tagset und Richtlinie für das Part-of-Speech-Tagging von Sprachdaten aus Genres internetbasierter Kommunikation. Guideline document.
- Michael Beißwenger, Sabine Bartsch, Stefan Evert, and Kay-Michael Würzner. 2016. EmpiriST 2015: A shared task on the automatic linguistic annotation of computer-mediated communication and web corpora. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X) and the EmpiriST Shared Task*, pages 44–56, Berlin. ACL.
- Devin Gaffney and J. Nathan Matias. 2018. Caveat emptor, computational social science: Large-scale missing data in a widely-published Reddit corpus. *PLOS ONE*, 13(7):1–13.
- Marco Lui and Timothy Baldwin. 2014. Accurate language identification of twitter messages. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*, pages 17–25, Gothenburg, Sweden. ACL.
- Thomas Proisl and Peter Uhrig. 2016. SoMaJo: State-of-the-art tokenization for German web and social media texts. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X) and the EmpiriST Shared Task*, pages 57–62, Berlin. ACL.
- Thomas Proisl. 2018. SoMeWeTa: A part-of-speech tagger for German social media and web texts. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 665–670, Miyazaki. ELRA.

GermEval 2019 Task 1: Hierarchical Classification of Blurbs

Steffen Remus, Rami Aly, Chris Biemann

Language Technology Group

Department of Informatics

Universität Hamburg, Germany

{remus,5aly,biemann}@informatik.uni-hamburg.de

Abstract

This paper presents the setup and outcome of the GermEval-2019 Task 1: Hierarchical Classification of Blurbs. A blurb is a short, occasionally advertorial, description of a book. The shared task consists of two subtasks: Task **A**) classification of blurbs exclusively into the most general categories, which can be considered to be a multi-label classification task, and Task **B**) hierarchical classification of blurbs into the entire hierarchy of categories, spanning a total of 343 different categories and sub-categories. During the test period, ten teams submitted 17 valid system solutions for Task A, and eight teams submitted 16 system solutions for Task B. For Task A, the best submission achieved a micro-F₁ score of 0.867, and for Task B the best submission achieved a micro-F₁ score of 0.677.

1 Introduction

Text classification (TC), as a sub-discipline in natural language processing (NLP), is an established task where many datasets for many target domains and challenges exist. Spam classification is probably the most well-known application of text classification algorithms. Here, the task is to classify messages (emails or short text messages) into two classes: spam (advertisements or any kind of harassment messages), or ham (relevant messages; [Gómez Hidalgo et al., 2006](#)¹). Due to the nature of this task and the fact that this resolves to binary text classification, it can be considered being solved with accuracy scores reaching 98+%, see e.g. ([Taheri and Javidan, 2017](#)). However, as more and more data become digitally available and people's time and convenience are growing in priority,

the demand for more, and finer-grained categories increases. Multi-class text classification gathered attention in this space (e.g. with the 20 Newsgroups dataset²), here the task is to classify an email (text and metadata) into one of 20 possible categories. As a next step, the multi-class text classification problem has been developed into a multi-label text classification problem, where a single sample can have one or multiple class labels. One of the popular datasets in this domain is the Reuters-21578 dataset³ ([Lewis, 1992](#)) which was superseded by the RCV1 dataset⁴ ([Reuters Corpus Volume 1; Lewis et al., 2004](#)), implementing a hierarchical structure on the classes. In hierarchical multi-label classification (HMC), labels are organized in a structured hierarchy, i.e. certain label combinations are irrelevant and should never be classified in conjunction ([Silla and Freitas, 2011](#)).

Hierarchical multi-label classification is not an entirely new challenge in the area of natural language processing ([Sun and Lim, 2001](#); [Silla and Freitas, 2011](#)), but with the increase of available data, especially on the web, the desire for more specific and specialized hierarchies increases. To cover this desire, and to foster research for algorithms dealing with hierarchically organized classes for the German Language in a real-world scenario, we present the *GermEval-2019 Task 1: Hierarchical Classification of Blurbs*, which includes two subtasks, where automatic systems have to infer: **A)** the **most general categories** of a book described by a blurb, and **B)** the **entire**

²<http://qwone.com/~jason/20Newsgroups/>

³<https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection>

⁴<http://www.daviddlewis.com/resources/testcollections/rcv1/>

¹<http://dcomp.sor.ufscar.br/talmeida/smsspamcollection/>

	Task A	Task B
#Teams:	10	8
#Submissions:	17	16
Best Team:	EricssonResearch	TwistBytes
Best Micro-F ₁ :	0.867	0.6767
Impr. over Baseline:	0.067	0.1428

Table 1: Quantitative details of submissions.

set of categories in the class hierarchy.^{5,6} Since a sample can belong to multiple classes on the same level, Task A can be considered as a standard multi-label classification task and a sub-problem of Task B, which is a hierarchical multi-label classification task. We compiled a hierarchical dataset of German blurbs by crawling the web pages of a major publisher and taking care of proper data cleaning and preparation.⁷ The details of the entire process, as well as various statistics, can be found in Section 3. For the shared task, we allowed three system submissions per team where eventually ten teams submitted 17 valid system solutions for Task A, and 16 valid system solutions were submitted by eight teams for Task B. Quantitative details of the test-phase submissions can be found in Table 1.

2 Prior Work

Text Classification Datasets:

The probably most well-known dataset with a hierarchical class label structure is the RCV1 (Reuters Corpus Volume 1; Lewis et al., 2004) dataset. It consists of roughly 800K documents categorized into several hierarchically structured category sets. However, the access to the dataset is limited and not freely usable by e.g. companies due to licensing. Lewis et al. (2004) distribute a term-document matrix where it has been ensured that the original data cannot be reconstructed. Therefore, many different variations of the original dataset have been created and used, and despite the wide acceptance of the dataset and extensive usage, it is difficult to directly compare

⁵GermEval is a series of shared task evaluation campaigns that focus on Natural Language Processing for the German language. The workshop is held in conjunction with the Conference on Natural Language Processing KONVENS 2019 in Erlangen/Nürnberg.

⁶<https://competitions.codalab.org/competitions/20139>

⁷We crawled the websites with the consent of the Random House publisher group.

results presented in scientific work due to the lack of availability of the standardized version.

Kowsari et al. (2017) introduced a hierarchically structured dataset for English, with a maximum depth of two, called the Web of Science Dataset: WOS-11967, WOS-46985 and WOS-5736 with 35, 134 and 11 categories and 7, 7 and 3 top-level categories respectively. However, in this dataset, every sample consists of exactly one parent-child label, which ultimately results in a single-label multi-class problem on the more specific category. This highly limits the diversity and complexity of the dataset and the underlying hierarchy. Several other large-scale datasets have been presented, e.g. (Kim et al., 2019; Mencía and Fürnkranz, 2010; Partalas et al., 2015). Some of these datasets consist of an extensive number of classes, up to several thousand. The classification of these datasets carry their very own challenges and are thus not further discussed here. In special application domains, such as the biomedical domain, more and more works include hierarchical structures in their data: e.g. Baker et al. (2015) introduced an annotated dataset based on the hallmarks of cancer (Baker et al., 2017) with a total of 37 classes and a hierarchy depth of 3 levels; Larsson et al. (2017) compiled a dataset for chemical risk assessment with a 32 classes and 5 levels.

Many freely accessible hierarchical datasets for the German language exist, however, no benchmark dataset has been established. For example, the OAI Protocol for Metadata Harvesting is a protocol designed to share metadata of catalogs and publications. However, the minimal requirements for expressing valid records are fairly loose and the practices of metadata management wildly differ across repositories. Attempts have been made to normalize OAI metadata records according to the hierarchical library taxonomy (Waltinger et al., 2009), called the Dewey Decimal Classification system. Multiple datasets of German patent collections have been created to classify these documents into the IPC taxonomy (Fall et al., 2004; Tikk et al., 2005).

HMC Approaches:

In text classification without hierarchical structures, neural architectures, especially *Convolutional Neural Networks* (CNNs) and different types of *Recurrent Neural Networks* (RNNs) (Goodfellow et al., 2016; Kim, 2014), most notably long short-term memory units (LSTMs,

Hochreiter and Schmidhuber, 1997) have shown to be highly effective. Cerri et al. (2014) use concatenated multi-layer perceptrons (MLP), where each MLP is associated with one level of the class hierarchy. In contrast, classifier chains (Read et al., 2011) employ binary classifiers for each category and propagate their predictions as a feature to the classifier for the child categories. However, this method is computationally expensive. Kowsari et al. (2017) use multiple concatenated deep learning architectures (CNN, LSTM, and MLP) for the WOS dataset – with a very shallow hierarchy and a fixed number of classes per example (one class label for each of the two hierarchy levels). Traditional classification approaches, such as e.g. KNN, Naïve Bayes or SVM, appear to fail to generalize adequately for large hierarchies (Kowsari et al., 2017). Summarizing, hierarchical multi-label classification brings research-worthy challenges, which motivated the conduction of this shared task.⁸

3 Dataset

In the following, we describe the preparation steps of the dataset, which are strongly in line with Aly et al. (2019).

3.1 Compiling the Dataset

The dataset is compiled using the openly available data of the (Bertelsmann) Random House (RH) webpage⁹. Random House is worldwide the largest publisher group and thus hosts an enormous body of books.

The German webpages of RH provide various meta information of books, such as a short description (the blurb), authorship information, title of the book, etc. (c.f. Figure 1). With the permission of the German RH division, we crawled¹⁰ the book listings, parsed the HTML code¹¹ and collected the following information that we considered to be relevant:

- title
- author(s)

⁸The official webpage of the shared task and respective data can be found at <https://www.inf.uni-hamburg.de/en/inst/ab/lt/resources/data/germeval-2019-hmc.html>.

⁹<https://www.randomhouse.de/>

¹⁰We crawled the webpages with Scrapy (<https://scrapy.org/>).

¹¹XPath and CSS were used to find and extract the necessary information.

- URL
- ISBN
- date of publication
- genres, i.e. categories
- info text, i.e. the blurb content

Other information such as *about the author*, or *reader's ratings* were ignored. The blurb of a book can be considered to be a short incentive description, which is occasionally advertorial (i.e. advertising and editorial) and thus clearly distinctive to a summary. Blurbs aim to bestir a potential reader to buy and read the book, they are thus designed to occasionally contain advertorial content. Each collected blurb can be considered unique, however, they might appear in similar forms, e.g. for books that are part of a series or are being republished as a new edition due to their success. Due to the extraction process of the sometimes noisy web data, anomalies such as missing author, missing blurb or incorrect publication date occurred infrequently for about 1% of the collected data and were thus accepted and kept in the dataset.

3.2 Category Refinement

The per-book extracted categories are lists of genres connected with their ancestor genres. Each book is thus categorized into a hierarchy. Still, this hierarchy contains ambiguities caused by the assignment of identical names to different categories allowing the formation of cycles as well as children to have multiple parents, e.g. *Science Fantasy* occurs as a subcategory of *Science Fiction* and *Fantasy*. Thus, we automatically renamed ambiguous categories by concatenating the category name to its parent's category name, and manually refined the extracted hierarchy further, which results in a tree-like categorical structure. Further, we manually checked all relations and merged or removed similar labels and removed categories that capture properties that do not rely on content but the shape or form of a book, e.g. categories such as *audiobook*, *ebook*, *hardcover*, *softcover*, etc. were removed. Finally, samples that have assigned category combinations that appear less than five times were also removed from the dataset.

3.3 Dataset Properties

The dataset follows the requirements as described in (Lewis et al., 2004): First, every book is as-

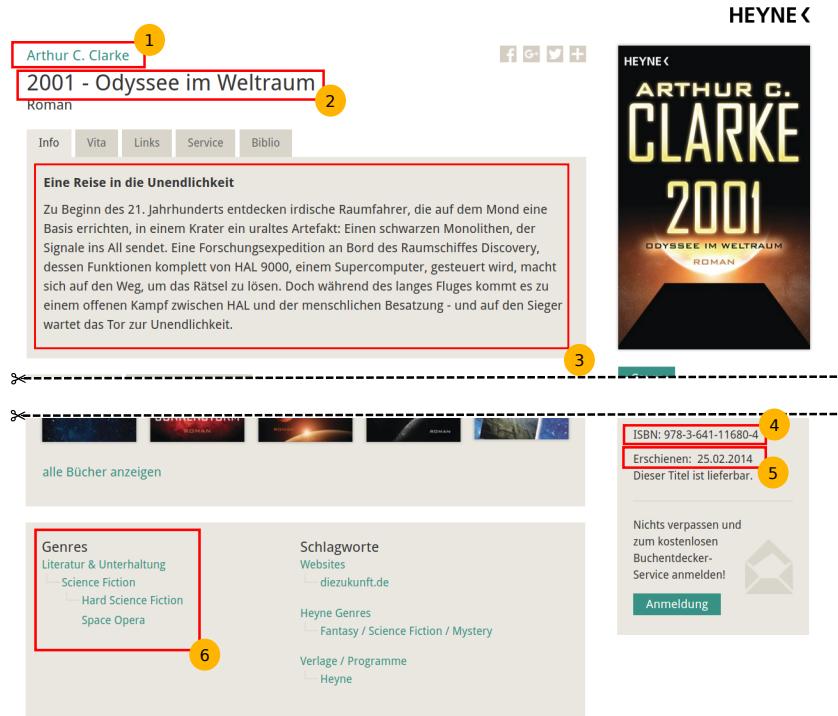


Figure 1: Snippet of website the data was collected from. The specific parts are highlighted in red boxes. Numbers indicate specific parts: (1) author name(s), (2) title, (3) blurb, (4) ISBN, (5) release date, (6) book's categories, displayed in a tree structure according to the underlying hierarchy. [The screenshot was taken in October 2018.]

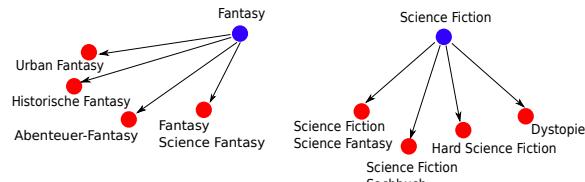


Figure 2: Excerpt of the hierarchy of categories. Colors indicate different levels in the hierarchy. The full hierarchy can be found in (Aly, 2018, p. 58).

signed at least one category, and second, every parent category in the path to the most general category of a book's most specific category is transitively assigned to it as well. In the dataset, the specified labels and the transitively assigned labels are distinguishable with the XML property `label` (value = `true` for most the specific label). Note that the most specific category of a book is not necessarily a leaf category in the hierarchy. For instance, the most specific category of a book could be *Children's Books*, although further child categories, such as *Middle-Grade books*, exist.

Figure 4 shows the frequency distribution of unique category combinations sorted by frequency

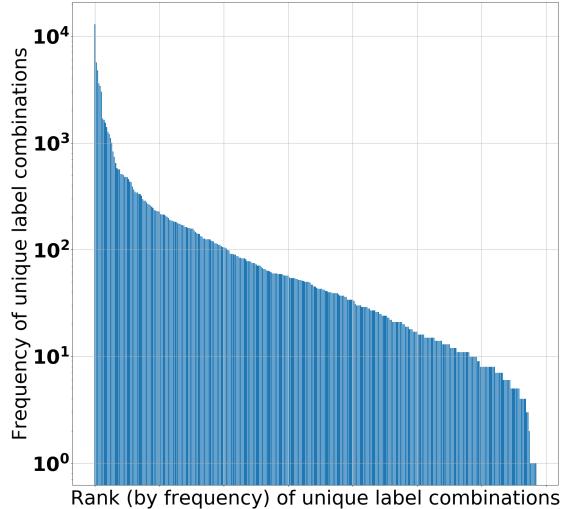


Figure 3: Frequency of category combinations (y-axis) in the entire dataset sorted by frequency rank (x-axis).

rank. As expected, few label combinations appear often and many label combinations appear rarely. The distribution of labels remains highly diverse with a total of 484 unique category combinations. Table 2 lists further important quantitative characteristics of the collected data such as the number of categories on each level of the hierarchy, etc.

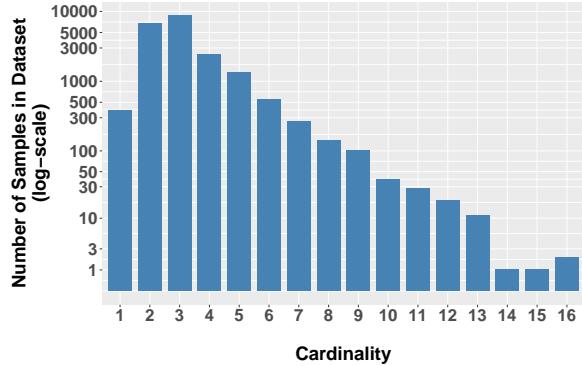


Figure 4: Distribution of the category cardinality per sample in the entire dataset.

For the task, we divided the dataset into three subsets: 70% training, 10% development and 20% test set ($\pm 0.2\%$ respectively). The dataset was split randomly with the constraint that every category in the development and test set occurs at least once in the training set. Additionally, maximally 2% of categories in the development and test set occur less than three times in the training set. While the test set is only used for the final evaluation of each system, the development set was used for benchmarking during the first evaluation phase. During the entire runtime of the task, participants were able to compare the performances of their systems via the CodaLab leaderboard for the development set. For the final evaluation phase, the development set labels have been supplied to the participants to allow a larger training set, and the CodaLab leaderboard was disabled for test set prediction submissions to avoid optimization on the test set.

4 Task Definition

The shared task contains two subtasks:

Task A: The task is to classify German books into one or multiple top-level categories. It can thus be considered a standard multi-label classification task. In total, there are eight top-level classes that can be assigned to a book: *Literatur & Unterhaltung* (*Literature & Entertainment*), *Ratgeber* (*Counsel*), *Kinderbuch & Jugendbuch* (*Books for Children and Young Adult Readers*), *Sachbuch* (*Nonfiction*), *Ganzheitliches Bewusstsein* (*Holistic Awareness*), *Glaube & Ethik* (*Belief & Ethics*), *Künste* (*Arts*), *Architektur & Garten* (*Architecture & Gardening*). The label distribution of these eight classes is highly imbalanced (cf. Figure 5).

#Samples	20,784
Average blurb length in tokens	94.67
Total number of categories	343
#Categories on level:	
1	8
2	93
3	242
#leaf nodes on level:	
1	0
2	51
3	242
Average branching factor	6.7 ± 4.97
Average branching factor on level:	
1	11.63 ± 6.39
2	5.76 ± 4.12
#Samples with labels of category on level:	
1	20,784
2	20,406
3	11,117
#Samples w/ cardinality (tlc [*]):	
1	19,422
2	1,260
3	97
4 (maximum cardinality)	5
#Samples w/ cardinality: see Figure 4 (maximum = 16)	
Average cardinality (tlc [*])	1.07 ± 0.28
Average cardinality	3.11 ± 1.37
#Distinct label combinations	484

Table 2: Quantitative characteristics of the dataset (*tlc: top-level-categories).

Task B: The second task is a hierarchical multi-label classification task where all categories of the hierarchy have to be assigned to a book. In total, 343 different classes are hierarchically structured, hence, not all combinations of categories are valid as defined by the hierarchy.

Submission Setup: The entire submission process was organized within the framework of a CodaLab competition¹². We limited the number of system submissions to three per team. The data release cycle went in three phases: In the first phase only a limited number of samples was released to familiarize with the structure of the dataset; in the second phase the training set with labels and the development set without labels were released and participants were able to submit their solutions for the development set to the CodaLab website; the

¹²<https://competitions.codalab.org/>

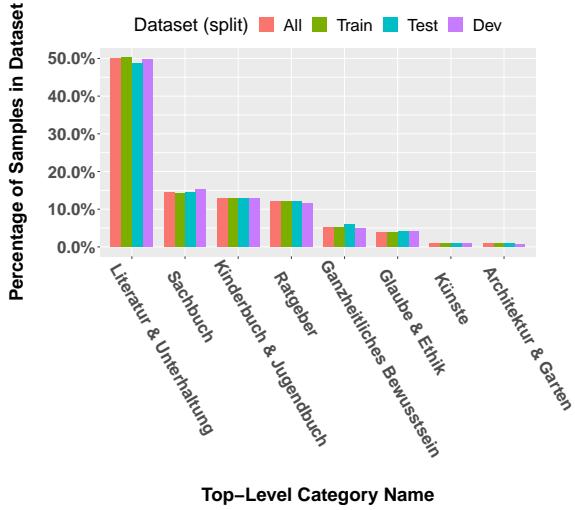


Figure 5: Top-level sample distribution.

third phase is the final test phase where the test set samples without labels and the labels for the development set samples were provided.

5 Systems

5.1 Organizer Systems

Baseline: SVM As a baseline method, we implement a traditional, non-hierarchical classifier using the *local approach* as described by Silla and Freitas (2011). We chose to use a linear SVM (Cortes and Vapnik, 1995) since it yielded good results in preliminary experiments. We exclusively use the blurb of a book to create features for the SVM and decided on minimal preprocessing, i.e. tokenization is performed using spaCy¹³ and stop words – as defined by spaCy – have been filtered. We then created a bag-of-word representation of unigrams and bigrams. Since the SVM is a binary classifier, we opted for a one-vs-all multi-label classification scenario, which was implemented using the scikit-learn library¹⁴. We use the standard value for the hyperparameter $C = 1$ and did not fine-tune it. Because predictions by independent classifiers do not necessarily lead to valid combinations as defined by the underlying hierarchy, we apply a post-processing step where we add missing parents of each predicted child label – recap that every child has an unambiguous parent. This process provides hierarchy-consistent label combinations but might lead to incomplete combinations because we do not add child labels

#Primary capsules	100
Convolution window size	50
Dimension of primary capsules	8
Dimension of class. capsules	8
Optimizer	Adam (Kingma and Ba, 2014)
Learning rate	0.002
#Epochs	10

Table 3: Hyper-parameter settings of the capsule network as found by non-exhaustive search.

for inner category nodes.

Contender: Capsule Networks Capsule networks have recently been shown to have advantages over traditional neural networks when confronted with structurally diverse categories and complex label co-occurrences (Aly et al., 2019; Zhao et al., 2018). For this reason, and the fact that the dataset is inherently unbalanced (as illustrated in Figure 3), we decided to employ a capsule network architecture from our previous work as a *contender system* for comparative reasons and out-of-competition. For the input, we tokenize the fields containing texts (title, author, and blurb) with spaCy and concatenate them. Tokens that appear only once in the dataset are replaced with a special unknown-token word. The sequence length has been limited to 100 tokens. We initialize an embedding layer with pre-trained fastText embeddings¹⁵ provided by Bojanowski et al. (2017) and adjust them during training. The structure of the capsule network follows tightly the implementation by Aly et al. (2019): Similar to CapsNet1 in (Xiao et al., 2018), our proposed system consists of four layers and every category in the hierarchy is associated with one class capsule in the network. As a post-processing step, we apply the same correction procedure as described above. Further hyper-parameter settings can be found in Table 3.

5.2 Submitted Systems

This section aims to give a quick overview of the different approaches used by the various teams for Task A and B, a short overview can be found in Table 4. We observe that the applied approaches can be grouped into two major groups, i.e. one focusing on the *local approach* where each node of the hierarchy is classified independently, here, mainly traditional classifiers are used, and one using the *global approach* where nodes are classified jointly

¹³<https://spacy.io/>

¹⁴<https://scikit-learn.org>

¹⁵<https://fasttext.cc/docs/en/pretrained-vectors.html>

Team	R _A	R _B	Classifier Approach	Text Features	Label (Post-) processing	Additional Data	Hierarchical Model Categorization
EricssonResearch (Umaashankar and Shanmugam S, 2019)	1	2	Conv Seq2Seq	fastText	random oversampling	—	global
TwistBytes (Benites, 2019)	2	1	one-vs-all SVM	TF-IDF n-grams + char n-grams	LCA	—	local per parent
DFKI-SLT (Ostendorff et al., 2019)	3	4	Transformer (BERT)	BERT	—	Wikidata KG Embeddings	global
Averbis (Genc et al., 2019)	6	3	Global CNN	fastText	T Criterion	—	global
Raghavan (K et al., 2019)	4	—	one-vs-all SVM	TF-IDF bi-grams	label count classifier	—	—
Fosil-hsmw (Bellmann et al., 2019)	5	—	SVM chain	GloVe + fastText	—	Author Database from RH	—
HSHL (Rother and Rettberg, 2019)	7	5	Logistic Regression + Naïve Bayes	TF-IDF uni-grams	limit by threshold	—	local
COMTRAVO-DS (Batista and Lyra, 2019)	8	6	Local CNNs	fastText	—	—	local
HUIU (Andresen et al., 2019)	9	—	one-vs-all SVM	BOW n-grams	limit by threshold	—	—
Baseline Contender	—	—	one-vs-all-SVM	BOW uni- & bi-grams	root path completion	—	local
	—	—	capsule networks	fastText	root path completion	—	global

Table 4: Overview of submitted approaches.

in the same model, here traditional and neural network classifiers are employed.

A variety of solution approaches have been submitted, 4 teams used SVM classifiers, where Fosil-hsmw opted for an RBF kernel and TwistBytes, HUIU, and Raghavan used a linear kernel function. HSHL decided to use a combined approach using Logistic Regression and Naïve Bayes, and 4 teams used neural network approaches, whereas 3 teams (EricssonResearch, COMTRAVO-DS, and Averbis) included convolutional layers in their architecture, and DFKI-SLT used an approach based on the transformer architecture (Vaswani et al., 2017), specifically BERT (Devlin et al., 2019). Whereas most teams used standard tokenization approaches such as spaCy, NLTK¹⁶, scikit-learn, etc., Raghavan use a Byte-Pair-Encoding (BPE) approach for tokenization. With those more general pieces of words, team Raghavan can build a more general vocabulary with reduced size. As text-representation within the classifier architecture, 4 teams decided to use traditional sparse representations in form of TF-IDF feature vectors (TwistBytes, Raghavan, HSHL) based on token-, POS-, or character n -grams and varying n (mostly $n = \{1, 2\}$). Fosil-hsmw, EricssonResearch, DFKI-SLT, COMTRAVO-DS, and Averbis relied on pre-trained embeddings, whereas Fosil-hsmw and EricssonResearch also trained embeddings on the provided blurbs.

fastText¹⁷ (Bojanowski et al., 2017) was mostly selected as the embedding framework of choice due to its ability to account for sub-word information and thus better handling of out-of-vocabulary words.

Other (provided) metadata processing, e.g. the number of authors, age of a book, gender of the author(s), ISBN-part splitting, etc., has been employed by several teams: Fosil-hsmw, EricssonResearch, DFKI-SLT, HUIU, and Raghavan. Further, external data was used by 2 teams: DFKI-SLT used knowledge graph embeddings based on Wikidata¹⁸, and Fosil-hsmw crawled the Random House website for additional author information to set up an author database and train task-specific embeddings.

Several teams studied the issue of label post-processing, i.e. the coherence of the hierarchy or more generally the number of labels to predict for a sample, by using several approaches: TwistBytes used a technique called LCA (Label Cardinality Adjustment; details can be found in their paper) for limiting the number of labels to predict, Averbis used a similar correction step as described in Section 5.1 named T-Criterion in order to correct non-connected child nodes, HSHL and HUIU used a threshold mechanism for the number of labels to predict (the threshold(s) were treated as a hyperparameter and optimized accordingly), and Raghavan used an independent prediction model for the number of labels. Motivated by the inherent imbalance of the sample size per

¹⁶<http://www.nltk.org/>

¹⁷<https://fasttext.cc>

¹⁸<https://wikidata.org>

label, EricssonResearch used random oversampling as a technique to balance the dataset.

6 Results and Discussion

6.1 Evaluation Metrics

Several metrics have been introduced to evaluate systems for hierarchical classification tasks, here, we use **micro-averaged** recall, precision, and F1-score and follow suggestions by [Silla and Freitas \(2011\)](#) and [Sorower \(2010\)](#). While macro-averaging, the respective scores are computed for each label individually and then averaged to produce a final single score; micro-averaged scores are computed globally for each metric over all instances. Thus, more frequent labels have a higher impact on the micro-averaged score, which essentially affects more general labels, since they appear more frequently in the dataset. Hence, we impose more importance on correct predictions on higher levels believing this yields to a more realistic scenario. ([Silla and Freitas, 2011](#)) suggest the use of micro-averaged scores for hierarchical classification tasks and even refer to them as hierarchical precision, recall, and F₁. However, these flat performance measures do not necessarily align with hierarchical ones, as shown in ([Brucker et al., 2011](#)), we thus additionally measure the hierarchical consistency score (HC) for Task B. This score measures the ratio of predictions made by the system that conform with the underlying label hierarchy, i.e. that all ancestors of a label are also assigned to the sample.

We further employ the exact match ratio or so-called *subset accuracy* (Acc) as described in ([Sorower, 2010](#)) because it captures how well labels are selected in relation to each other. In contrast to the F₁-score, which takes partially correct classifications into account, the subset accuracy is a very strict metric as there is no distinction between partially correct classification and completely incorrect classifications.

6.2 Quantitative evaluation

The extensive list of results during the test phase and the post-evaluation phase is shown in the appendix [A](#) and [B](#). The following analysis is based only on the results of the best system submitted by each team during the test phase.

Task A: Scores of the best system submission from each team for Task A are listed in Table [6](#). The best performing system achieved a

micro-F₁ score of 0.867 and was submitted by EricssonResearch¹⁹. Besides, this system has also achieved the highest subset accuracy with a significant margin to the second-highest score.

Further analysis of the scores for each top-level category shows that the system by EricssonResearch performed especially well on categories with the fewest samples in the dataset, i.e. *Architektur & Garten (Architecture and Gardening)* and *Künste (Arts)* as can be seen in Table [5](#). In contrast, our Baseline system performs the worst for these classes and lacks behind significantly to all submissions. For categories with a high number of examples such as *Literatur & Unterhaltung (Literature & Entertainment)*, all submitted systems perform equally, which indicates that the main challenge for Task A might be data sparsity. EricssonResearch was the only team that explicitly addressed this issue by using random oversampling.

Task B: Results for Task B are listed in Table [7](#). Team TwistBytes submitted the system with the highest F₁ score of 0.6767. The subset accuracy score of 0.3791 of the system by EricssonResearch (2nd rank) is particularly interesting, outperforming all other teams by at least 11%. Regarding hierarchy conformity (HC), five out of six systems have a perfect score concerning the inherent category hierarchy (HC). Notably, the system submitted by DFKI-SLT has an almost perfect hierarchy consistency (HC) score although they do not directly encode any hierarchy information within their model. Again, the Baseline system was outperformed by a large margin, scoring lowest of all systems in terms of recall, but surprisingly also achieving the highest precision score.

The capsule network (contender) performs in the mid-range, while the only other global approach that outperforms the capsule network is by EricssonResearch.

Further analysis of F₁ scores on each hierarchy level shows a performance decline throughout all systems for categories on deeper, and thus sparser, levels (c.f. Figure [6](#) (a) and (b)).

¹⁹Note that team Raghavan submitted improved results in the post-evaluation phase that beat the best results of the test phase.

Team	Literatur & Unterhaltung	Sachbuch	Kinderbuch & Jugendbuch	Ratgeber	Ganzheitliches Bewusstsein	Glaube & Ethik	Architektur & Garten	Künste
EricssonResearch	0.93	0.75	0.88	0.79	0.78	0.75	0.77	0.85
twistbytes	0.92	0.76	0.87	0.79	0.80	0.78	0.71	0.74
DFKI-SLT	0.93	0.78	0.84	0.79	0.79	0.73	0.69	0.81
Raghavan	0.93	0.75	0.87	0.79	0.74	0.74	0.65	0.65
Fosil-hsmw	0.92	0.71	0.84	0.73	0.73	0.74	0.71	0.77
Averbis	0.92	0.71	0.82	0.73	0.77	0.74	0.56	0.68
HSHL	0.90	0.72	0.76	0.74	0.74	0.72	0.65	0.62
Comtravo-DS	0.90	0.71	0.78	0.76	0.74	0.73	0.65	0.67
HUIU	0.89	0.70	0.74	0.73	0.71	0.68	0.61	0.73
Contender	0.91	0.71	0.83	0.76	0.78	0.77	0.71	0.77
Baseline	0.90	0.68	0.69	0.72	0.69	0.63	0.34	0.45
# Samples in test set	2182 (49%)	650 (14%)	575 (13%)	536 (12%)	262 (6%)	183 (4%)	44 (1%)	38 (<1%)

Table 5: F₁ scores for top-level categories for Task A.

Rank	best System by Team	Acc	Precision	Recall	F ₁
1	EricssonResearch	.84	.89	.84	.87
2	TwistBytes	.79	.87	.86	.86
3	DFKI-SLT	.82	.88	.85	.86
4	Raghavan	.83	.88	.84	.86
5	Fosil-hsmw	.79	.84	.83	.84
6	Averbis	.79	.86	.81	.83
7	HSHL	.77	.82	.82	.82
8	Comtravo-DS	.72	.81	.83	.82
9	HUIU	.76	.81	.81	.81
	Contender	.74	.82	.85	.84
	Baseline	.71	.86	.75	.80

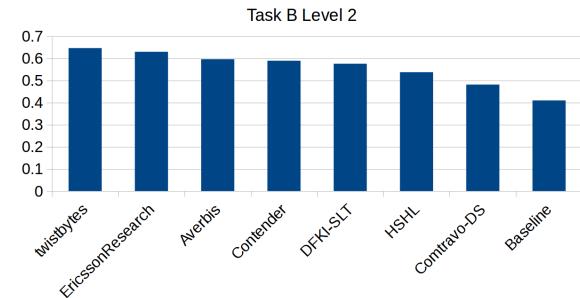
Table 6: Results for Task A of participating teams. Only the best performing system per team is listed. Scores are micro-averaged.

Rank	Model	Acc	Precision	Recall	F ₁	HC
1	Twistbytes	.25	.71	.65	.68	1
2	EricssonResearch	.38	.74	.62	.67	1
3	Averbis	.27	.68	.61	.64	1
4	DFKI-SLT	.21	.78	.52	.62	.97
5	HSHL	.26	.72	.54	.62	1
6	Comstravo-DS	.19	.70	.53	.60	1
	Contender	.25	.76	.56	.64	1
	Baseline	.15	.85	.39	.53	1

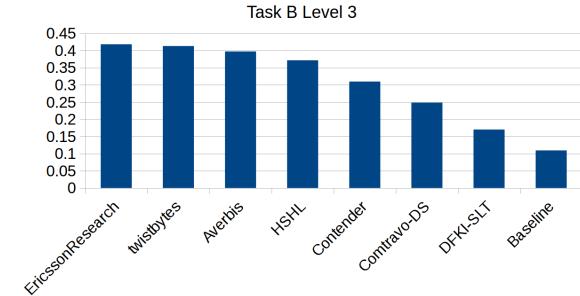
Table 7: Results for Task B of all participating systems. Only the best performing system is listed. Illustrated scores are micro-averaged.

7 Summary

We presented the summary report of the *GermEval-2019 Task 1: Hierarchical Classification of Blurb*s which included two sub-tasks: classification of categories of different granularities. As part of this shared task, participants were provided with a dataset consisting of blurbs including metadata in German of around 20K books. The shared task consisted of three phases: the first phase was designed to familiarize with the task and the data, the second phase provided the training data and a platform to compare the performance of submissions on the held-out validation set, and the third phase provided access to the



(a) F₁ scores on categories that are on the second level of the label hierarchy.



(b) F₁ scores on categories that are on the third level of the label hierarchy.

Figure 6: Performance report on different levels of the hierarchy.

validation data for additional training and disabled performance comparisons on the held-out test set for fairness purposes. System submissions cover a variety of approaches to deal with the category hierarchy: three systems (+ baseline) were designed using the local approach, either by learning one model (SVM or CNN) per parent node or per level. Four (+ contender) systems employed the global approach: three teams use CNNs and one uses transformer networks with a linear decoder on top. Most systems incorporated the hierarchy directly into their system or employed a post-processing step to adjust predictions. While

some of the top-performing teams employed deep neural network architectures either for learning a representation of blurbs or for the classification task itself, well adjusted and fine-tuned traditional classifiers have shown competitive results.

Acknowledgments

We would like to thank the publisher group Random House for their permission to crawl their website and to make the dataset publicly available. We also congratulate all participants on their achievements and appreciate the diversity of the approaches. This work was partially supported by the Forum 4.0 project, funded by Hamburg's BWFG.

References

- Rami Aly. 2018. Hierarchical writing genre classification with neural networks. B.Sc. Thesis, Universität Hamburg, Germany.
- Rami Aly, Steffen Remus, and Chris Biemann. 2019. [Hierarchical multi-label classification of text with capsule networks](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 323–330, Florence, Italy. Association for Computational Linguistics.
- Melanie Andresen, Melitta Gillmann, Jowita Grala, Sarah Jablotschkin, Lea Röseler, Eleonore Schmitt, Lena Schnee, Katharina Straka, Michael Vauth, Sandra Kübler, and Heike Zinsmeister. 2019. The HUIU contribution to the GermEval 2019 shared task 1. In *GermEval 2019, 15th Conference on Natural Language Processing (KONVENS 2019)*, Erlangen, Germany.
- Simon Baker, Imran Ali, Ilona Silins, Sampo Pyysalo, Yufan Guo, Johan Höglberg, Ulla Stenius, and Anna Korhonen. 2017. Cancer hallmarks analytics tool (chat): a text mining approach to organize and evaluate scientific literature on cancer. *Bioinformatics*, 33(24):3973–3981.
- Simon Baker, Ilona Silins, Yufan Guo, Imran Ali, Johan Höglberg, Ulla Stenius, and Anna Korhonen. 2015. Automatic semantic classification of scientific literature according to the hallmarks of cancer. *Bioinformatics*, 32(3):432–440.
- David S. Batista and Matti Lyra. 2019. COMTRAVO-DS team at GermEval 2019 task 1 on hierarchical classification of blurbs. In *GermEval 2019, 15th Conference on Natural Language Processing (KONVENS 2019)*, Erlangen, Germany.
- Franz Bellmann, Lea Bunzel, Christoph Demus, Lisa Fellendorf, Olivia Graupner, Qiuyi Hu, Tamara Lange, Alica Stuhr, Jian Xi, Michael Spranger, and Dirk Labudde. 2019. Multi-label classification of blurbs with SVM classifier chains. In *GermEval 2019, 15th Conference on Natural Language Processing (KONVENS 2019)*, Erlangen, Germany.
- Fernando Benites. 2019. TwistBytes - hierarchical classification at GermEval 2019: walking the fine line (of recall and precision). In *GermEval 2019, 15th Conference on Natural Language Processing (KONVENS 2019)*, Erlangen, Germany.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. [Enriching word vectors with subword information](#). *Transactions of the Association for Computational Linguistics*, 5(1):135–146.
- Florian Brucker, Fernando Benites, and Elena Sapozhnikova. 2011. An empirical comparison of flat and hierarchical performance measures for multi-label classification with hierarchy extraction. In *Knowledge-Based and Intelligent Information and Engineering Systems*, pages 579–589, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Ricardo Cerri, Rodrigo C. Barros, and André C.P.L.F. de Carvalho. 2014. [Hierarchical multi-label classification using local neural networks](#). *Journal of Computer and System Sciences*, 80(1):39 – 56.
- Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, MN, US.
- C.J. Fall, A. Törcsvári, P. Fiévet, and G. Karetka. 2004. [Automated categorization of german-language patent documents](#). *Expert Systems with Applications*, 26(2):269 – 277.
- Erdan Genc, Louay Abdelgawa, Viorel Morari, and Peter Kluegl. 2019. Convolutional neural networks for classification of German blurbs. In *GermEval 2019, 15th Conference on Natural Language Processing (KONVENS 2019)*, Erlangen, Germany.
- José María Gómez Hidalgo, Guillermo Cajigas Bringas, Enrique Puertas Sánz, and Francisco Carrero García. 2006. Content based SMS spam filtering. In *Proceedings of the 2006 ACM Symposium on Document Engineering*, pages 107–114.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. [Http://www.deeplearningbook.org](http://www.deeplearningbook.org).
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

- Raghavan A K, Venkatesh Umaashankar, and Gautham Krishna Gudur. 2019. Label frequency transformation for multi-label multi-class text classification. In *GermEval 2019, 15th Conference on Natural Language Processing (KONVENS 2019)*, Erlangen, Germany.
- Kang-Min Kim, Yeachan Kim, Jungho Lee, Ji-Min Lee, and SangKeun Lee. 2019. From small-scale to large-scale text classification. In *The World Wide Web Conference, WWW ’19*, pages 853–862, New York, NY, USA.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar.
- Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, Banff, Canada.
- Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes. 2017. HDLTex: Hierarchical deep learning for text classification. In *IEEE International Conference on Machine Learning and Applications*, pages 364–371, Cancún, Mexico.
- Kristin Larsson, Simon Baker, Ilona Silins, Yufan Guo, Ulla Stenius, Anna Korhonen, and Marika Berglund. 2017. Text mining for improved exposure assessment. *PloS one*, 12(3):1–21.
- David D. Lewis. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In *Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–50.
- David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5(Apr):361–397.
- Eneldo Loza Mencía and Johannes Fürnkranz. 2010. Efficient multilabel classification algorithms for large-scale problems in the legal domain. In *Semantic Processing of Legal Texts*, pages 192–215. Springer.
- Malte Ostendorff, Peter Bourgonje, Maria Moritz, Julián Moreno-Schneider, and Georg Rehm. 2019. Enriching BERT with knowledge graph embeddings for document classification. In *GermEval 2019, 15th Conference on Natural Language Processing (KONVENS 2019)*, Erlangen, Germany.
- Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artieres, George Palouras, Eric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Galinari. 2015. Lshtc: A benchmark for large-scale text classification. *ArXiv:1503.08581*.
- Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359.
- Kristian Rother and Achim Rettberg. 2019. Logistic regression and naive bayes for hierarchical multi-label classification at GermEval 2019 - task 1. In *GermEval 2019, 15th Conference on Natural Language Processing (KONVENS 2019)*, Erlangen, Germany.
- Carlos N. Silla and Alex A. Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72.
- Mohammad S. Sorower. 2010. A literature survey on algorithms for multi-label learning. Oregon State University, Corvallis, OR, USA.
- Aixin Sun and Ee-Peng Lim. 2001. Hierarchical text classification and evaluation. In *Proceedings of the 2001 IEEE International Conference on Data Mining, ICDM ’01*, pages 521–528, San Jose, CA, USA.
- Rahim Taheri and Reza Javidan. 2017. Spam filtering in sms using recurrent neural networks. In *2017 Artificial Intelligence and Signal Processing Conference (AISP)*, pages 331–336, Shiraz, Iran.
- Domonkos Tikk, György Biró, and Jae Dong Yang. 2005. Experiment with a Hierarchical Text Categorization Method on WIPO Patent Collections, pages 283–302. Boston, MA, USA.
- Venkatesh Umaashankar and Girish Shanmugam S. 2019. Multi-label multi-class hierarchical classification using convolutional Seq2Seq. In *GermEval 2019, 15th Conference on Natural Language Processing (KONVENS 2019)*, Erlangen, Germany.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.
- Ulli Waltinger, Alexander Mehler, Mathias Lösch, and Wolfram Horstmann. 2009. Hierarchical classification of OAI metadata using the DDC taxonomy. In *Advanced Language Technologies for Digital Libraries*, pages 29–40, Trento, Italy.
- Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. MCapsNet: Capsule network for text with multi-task learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4565–4574, Brussels, Belgium.
- Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating capsule networks with dynamic routing for text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language*, pages 3110 – 3119, Brussels, Belgium.

A Full Results Task A

Rank	Team	System	F1	Precision	Recall	Support	Phase
1	Raghavan	SVM-BPEB	0.88	0.90	0.86	1.00	post-eval
2	EricssonResearch	CC_a_fconv_b_A6C1Y	0.87	0.89	0.84	0.98	post-eval
3	EricssonResearch	fconv_A6C1Y	0.87	0.89	0.84	1.00	test
4	twistbytes	baseline_lca	0.86	0.85	0.88	0.99	post-eval
5	twistbytes	sklearn_hier_threshold_and_roots_baseline_thresholding	0.86	0.86	0.86	0.96	test
6	DFKI-SLT	full	0.86	0.88	0.85	1.00	test
7	Raghavan	SVM-BPEB	0.86	0.88	0.84	1.00	test
8	twistbytes	baseline_0_25	0.86	0.82	0.90	1.00	post-eval
9	DFKI-SLT	text-only	0.86	0.87	0.84	1.00	test
10	EricssonResearch	fconv_F8V17	0.85	0.88	0.83	1.00	test
11	knowcup	DL_single_test	0.84	0.85	0.84	1.00	test
12	DFKI-SLT	full2	0.84	0.87	0.81	1.00	test
13	LT	Contender	0.84	0.82	0.85	1.00	test
14	fosil-hsmw	SVM_ECC	0.84	0.84	0.83	1.00	test
15	Averbis	BOHB_CNN	0.83	0.86	0.81	0.98	test
16	twistbytes	sklearn_hier_threshold	0.83	0.91	0.76	0.85	test
17	HSHL	LogisticRegression_NaiveBayes1	0.82	0.82	0.82	1.00	test
18	Comtravo-DS	local_clf_logit_cnn	0.82	0.81	0.83	0.94	test
19	HSHL	LogisticRegression_NaiveBayes2	0.82	0.82	0.81	1.00	test
20	HUTU	mult	0.81	0.81	0.81	1.00	test
21	LT	Baseline_wo_correction	0.80	0.86	0.75	0.88	test
21	LT	Baseline	0.80	0.86	0.75	0.88	test
22	Comtravo-DS	global_clf_cnn	0.78	0.78	0.78	0.99	test
23	EricssonResearch	fconv_4LYFP_7EKHC_WNG1A	0.66	0.68	0.64	1.00	test

B Full Results Task B

Rank	Team	System	F1	Precision	Recall	Support	Phase
1	twistbytes	sklearn_hier_threshold_and_roots_baseline_thresholding	0.68	0.71	0.65	0.98	test
1	twistbytes	sklearn_hier_threshold	0.68	0.71	0.65	0.98	test
2	EricssonResearch	fconv_A6C1Y	0.67	0.74	0.62	1.00	test
2	EricssonResearch	CC_a_fconv_b_A6C1Y	0.67	0.74	0.62	1.00	post-eval
3	EricssonResearch	fconv_F8V17	0.66	0.72	0.60	1.00	test
4	knowcup	DL_single_test	0.65	0.75	0.58	1.00	test
5	Averbis	BOHB_CNN	0.64	0.68	0.61	1.00	test
6	LT	Contender	0.64	0.75	0.56	1.00	test
7	DFKI-SLT	full	0.62	0.78	0.52	1.00	test
7	DFKI-SLT	full2	0.62	0.78	0.52	1.00	test
8	HSHL	LogisticRegression_NaiveBayes1	0.62	0.72	0.54	1.00	test
9	HSHL	LogisticRegression_NaiveBayes2	0.61	0.74	0.51	1.00	test
10	Comtravo-DS	local_clf_logit_cnn	0.60	0.70	0.53	0.94	test
11	DFKI-SLT	text-only	0.58	0.72	0.49	1.00	test
12	Comtravo-DS	global_clf_cnn	0.54	0.57	0.52	1.00	test
13	LT	Baseline	0.53	0.85	0.39	0.88	test
14	LT	Baseline_wo_correction	0.53	0.85	0.39	0.88	test
15	EricssonResearch	fconv_4LYFP_7EKHC_WNG1A	0.48	0.58	0.42	1.00	test
16	twistbytes	baseline_0_25	0.45	0.82	0.31	1.00	post-eval
17	twistbytes	baseline_lca	0.44	0.85	0.30	0.99	post-eval
18	twistbytes	thresholding	0.44	0.86	0.30	0.96	test
19	Raghavan	SVM-BPEB	0.39	0.70	0.27	1.00	post-eval
20	NoTeam	GRU_Attention_ensemble1	0.33	0.42	0.28	1.00	test

Multi-Label Classification of Blurbs with SVM Classifier Chains

**Franz Bellmann, Lea Bunzel, Christoph Demus, Lisa Fellendorf, Olivia Gräupner,
Qiuyi Hu, Tamara Lange, Alica Stuhr, Jian Xi, Dirk Labudde, Michael Spranger**

University of Applied Sciences Mittweida

Technikumplatz 17

09648 Mittweida

spranger@hs-mittweida.de

Abstract

Due to the increasing digitalisation multi-label classification gains in importance in many areas. In this paper we propose a method to classify blurbs into eight basic book genre using an ensemble of classifier chains composed of radial support vector machines using word embeddings and author information as features. Five models were tested using different implementations and features, as well as different numbers of chains. The best model reached a performance of a micro average F1 of 0.841.

1 Introduction

The increasing digitalisation often requires the integration of data and print media. In most cases, the first step is the classification of the datasets, or more specifically the documents, into a taxonomy, for example in order to assign these to different fields. For instance, before a bank approves a credit request, all the information given by the applicant has to be assigned to different categories to assess whether all required documents have been handed in or, when necessary, to inform the responsible official. Additional applications might be in forensics to classify evidential documents or in the library system. For the latter, the task is to sort the books into a thematic library taxonomy based on the short summary given on the back of a book's cover (blurbs). The difficulty, compared to a simple classification task, is the multinomially mapping of the books to the labels in a taxonomy, meaning each book can be assigned to more than one category which can belong to different taxonomy levels (Remus et al., 2019).

The GermEval 2019 shared task addresses this task for the German language with two subtasks, whereas the second one focuses on the different taxonomy levels. The data consists of blurbs of German books, which are provided by the publisher Random House. These blurbs and several

meta information for instance the title and author have to be categorized into the most common writing genres and subgenres of German literature. For the first task these are only the taxonomy entries of the first level, namely: *Literatur & Unterhaltung*, *Ratgeber*, *Kinderbuch & Jugendbuch*, *Sachbuch*, *Ganzheitliches Bewusstsein*, *Glaube & Ethik*, *Künste*, and *Architektur & Garten*. In this paper an approach based on chained SVM models is presented and tested for the first task. The paper is organized as follows: First, some related work is presented in Section 2. Then an overview is given of the data and the methods in Sections 3 and 4. The results are presented in Section 5 before we conclude with Section 6.

2 Related Work

In the last two decades a lot of research has been conducted in the field of multi-label text classification whereas over time research has focused on several approaches. The most obvious approach is to adapt classifiers, such as kNN (Zhang and Zhou, 2005) or neural networks, to the multi-label task. Yet, typically, such classifiers have some shortcomings such as a high algorithmic complexity, which may lead to high computational costs. Another possibility is the transformation of the problem into several binary classification problems, also known as binary relevance approach. In this case two approaches exist. The one-versus-all approach trains one binary classifier per label and the all-versus-all one for each possible label combination. While the complexity of the one-versus-all approach grows linearly, the complexity of the all-versus-all approach increases approximately quadratically. However, the disadvantage of the one-versus-all approach is that it does not consider possible label correlations. One way to address this problem is to build a final classifier

with two stages. In the first stage each documents is classified using the binary classifier and then, in the second stage, the decisions are added to the input vector before the classifier is trained again, so that, in fact, the classifiers do not work independently anymore (Godbole and Sarawagi, 2004). Based on the aforementioned paper Read et al. (2009; 2011) developed classifier chains and subsequently ensembles of them which address the problem that different classifier orders result in different decisions.

Another important part is the representation of the documents. Usually, they are modelled as a term frequency vector (bag of words) in a Vector Space as described in Salton et al. (1975), yet, the resulting document vectors are high dimensional and sparse. Subsequently, vectors representing words and their relations in low dimensional space can be used as introduced in Mikolov et al. (2013a). Additionally, methods like Global Vectors (Pennington et al., 2014) and FastText (Joulin et al., 2017) were developed but with these approaches alone complete texts cannot be represented as low dimensional vectors. Addressing this problem Mikolov et al. (2013b) described that the addition of this word vectors produces meaningful results. Furthermore Yin and Jin (2015) hypothesize that the sum of word vectors of all words in a document results in a meaningful document vector. Although they apply the idea only to skip gram models Chilakapati (2018) generalized this to all word embeddings.

3 Data

The data used in this paper was provided by the organizers of the first task for the GermEval2019 and consisted of blurbs from 20,784 books from the publisher Random House. The models described in this paper were trained on the training set containing 14,548 blurbs and evaluated with the validation set (2079 blurbs). For the submitted model both, the training set as well as the validation data set, were used.

Each document consists of title, blurb, author, URL, ISBN, release date and associated labels. As can be seen in Table 1 the dataset is unbalanced with the category *Literatur & Unterhaltung* with the largest amount of books having 7817 books compared to 128 in the category *Architektur & Garten*.

Furthermore, some anomalies could be ob-

served while exploratively analysing the data, which, however, only concerned about one percent of the data. For example, for some books no authors were available, while for others the blurbs were missing. In the second case, the data samples would have a null vector as a document vector and, consequently, would not have been assigned to a category. However, as additional author information was used as one feature some documents with a missing body were simply assigned to the author’s genre. Books with no author information were handled in the same way as those books for which no information was available in the created author database (see Section 4.3).

Additionally, a few special books were found in the dataset. Their ISBN starts with a four, whereas usually the ISBN begins with the digit nine. After a short overall inspection concerning these objects, it was discovered that a few so called fan products were placed in the data set. The first idea was to delete them, but because of an existing body, it was decided to keep them. In fact, it turned out that most of them were assigned to the correct genre. Another problem was that the category *Ratgeber* has no equivalent category on www.randomhouse.de. This problem could not be solved. As a result the category *Ratgeber* was not taken into account for the additional feature based on the writing genre of the authors.

Genre	# blurbs
Literatur & Unterhaltung	7817
Sachbuch	2201
Kinderbuch & Jugendbuch	1987
Ratgeber	1862
Ganzheitliches Bewusstsein	803
Glaube & Ethik	598
Künste	146
Architektur & Garten	128

Table 1: Number of blurbs in each genre.

4 Methods

To solve the given classification task, an ensemble classifier chain as described by Read et al. (2009; 2011) was used. Global Vector as well as FastText representations of the texts combined with the information in which genre the respective author mostly publishes their work were considered as features. Due to the unbalanced nature of the

data the micro average F1-measure was used as the main evaluation criterion.

4.1 Encoding multi-class labels

For this task each possible category combination a book can belong to is encoded in one single number in order to handle these multi label categories easier. Hence, the category vector is considered as a bit pattern of the size eight. Each position in this pattern is related to one genre, as shown in Table 2.

Encoding	Genre
1	Literatur & Unterhaltung
2	Sachbuch
4	Kinderbuch & Jugendbuch
8	Ratgeber
16	Ganzheitliches Bewusstsein
32	Glaube & Ethik
64	Künste
128	Architektur & Garten

Table 2: Encoding for each genre.

Consequently, multi-label categories can be represented by adding up those bit values, so every combination has its own unique number. For example, if a document was classified as *Literatur & Unterhaltung* and *Glaube & Ethik* at the same time, the binary code would be 00100001 (in decimal: $1 + 32 = 33$). Therefore, its (multi label) category would be 33.

4.2 Preprocessing

As the data was provided by the organizers in an XML format, in a first step, it was converted into CSV data sets. Then, the data was filtered in order to get rid of the fine-grained categories. Furthermore, all columns except of *title*, *author* and *body* were removed. Afterwards, the blurbs contained in the body were tokenized into words and normalized. This included the conversion to lower case as well as lemmatizing and POS tagging using TreeTagger (Schmid, 1994). For further process steps only adjectives, nouns, verbs and adverbs were used.

4.3 Feature modelling

Vectorization

Next, the blurbs were vectorized, by converting each blurb into a low dimensional document vector. To do so, a pre-trained Global Vector (GloVe)

with 300 dimensional word vectors based on a bag of words (BOW) containing a corpus with about 850,000 words of the German Wikipedia was downloaded from Pietsch et al. (2018). In order to create low dimensional and non-sparse document vectors the method described in Yin and Jin (2015) was used. The authors show that taking word vectors into account leads to more meaningful results which is basically the idea of word embeddings.

Subsequently, each document representation is simply the sum of these vectors weighted by the term frequency of the respective word, as show in Equation 1, were n is the number of words in the corpus, t_j is the frequency of each word w_i in the specific document, and v_j represents the word vector of each word w_i .

$$\vec{d}_i^* = \sum_{j=1}^n (t_j \cdot v_j) \quad (1)$$

For comparison, we built a new 230 dimensional FastText model (continues bag of words), which was trained on about 29,000 blurbs, consisting of the blurbs in the provided data and additional blurbs crawled from www.randomhouse.de.

At the end, the document vectors were standardized to the euclidean length 1.

Including author information

As mentioned above, information about the author was included as an additional feature. The necessary information was crawled from the publisher's website www.randomhouse.de. More precisely, information about 15,717 authors was crawled to find out how many books an author wrote in each of the eight categories. Then, this information was transformed into one vector a for each author containing eight elements a_1, \dots, a_8 , one for each category, each including the publishing frequency of the author in this category. As already discussed before, no information was available for the category *Ratgeber*. Further, each vector was normalized in order to get comparable results as well as weighted to achieve higher values if the author is very active as shown in Equation 2.

$$a'_i = \frac{a_i}{\sum_{j=1}^n a_j} \log(a_i + 1) \quad (2)$$

Finally, the author vector was appended to the document vector to include it in the model.

4.4 Classifier Chains

Classifier chains are first described in Read et al. (2009) and consist of several binary classifiers linked together. It is a method which turns a multi-label classification problem into n binary ones, whereas n is the number of possible categories. In this specific case, there is one binary classifier for each of the eight categories. The advantage of this method is that the classifiers do not work independently, but take into account the other classifiers' results. Thereby, correlations between the labels will have an influence on the final result. This is especially valid if there exist interdependencies between categories as is the case in highly branched taxonomies. Even though this is obviously not the case for the coarse classification task, it can not be completely excluded and it is definitely relevant for the fine-grained classification task (classification including subgenres). The models in this paper use these classifier chains as described in Read et al. (2009; 2011) and an overview is given in Figure 1.

The input for the classifier chain is a document's feature vector \vec{f} containing the word features w_i from the document vector \vec{d}_i^* and the author features a_1, \dots, a_8 . In a specified order each binary classifier predicts one category and forwards $[\vec{f}, c_i]$ as an extended document vector, whereas c_i is the predicted result of this classifier. The additional dimensions represent the multi-label classification of the document.

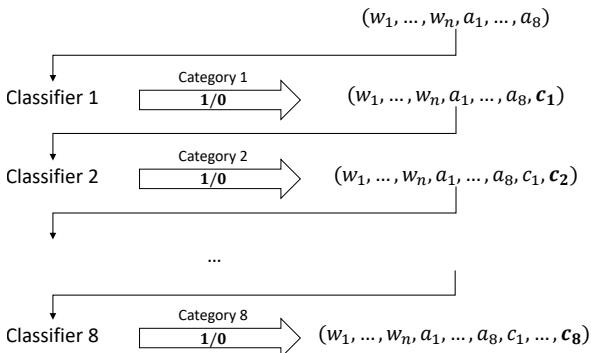


Figure 1: Structure of a classifier chain. The input for the chain is a document vector consisting of word features w_i and author features a_i . Each linked classifier adds its decision as feature c_i to that vector. The resulting vector serves as the input for the next classifier.

Basically, the order of the classifiers in the chain can be chosen arbitrarily or randomly. If there is an inherent order between categories then this order should be resembled in the order of the classi-

fiers. As already described there is no such inherent order in the coarse classification task. Therefore, a random order was chosen. Any hidden dependencies can be considered by classifying repeatedly in different orders as described in Section 4.5.

The binary classifiers represent the core of a classifier chain. For the used models a Support Vector Machine with a one-versus-all technique was chosen because it works well with high dimensional vectors. The Support Vector Machine finds the optimal hyper-plane in the feature space in order to separate the categories and then classifies new vectors by mapping them into the feature space (Lee et al., 2011).

In this study, two different implementations of Support Vector Machines were tested. Both were used with an Radial Basis Function kernel which maps the vectors in a non linear way into the feature space. Therefore, it can handle non linear correlations between the features. Additionally, the classes were weighted because the dataset is very unbalanced (Hsu et al., 2016).

Both implementations differ in the way to train the Support Vector Machine. Using LibSVM (SVM-C) (Chang and Lin, 2001) requires the user to set the parameters C and γ manually. In this study the standard parameters $C = 1$ and $\gamma = 1/|\vec{f}|$ were chosen. In contrast, the caret package (version 6.0-84) for R implements a radial Support Vector Machine that tries to optimize its parameters using a given number (in this case 20) of randomly chosen parameter sets (Kuhn, 2019).

4.5 Ensemble Classifier Chains

The ensemble classifier chains are a method to overcome limitations of classifier chains and to improve their performance. As mentioned before, the performance of a classifier chain may depend on the order of the single classifiers and may lead to different results. In a learning ensemble this problem is minimized by grouping together a number of classifier chains each with a different order. It was used as described in Read et al. (2009; 2011).

Subsequently, a function is needed to determine the number of chains that need to come to the same voting in order to determine the overall category for a given document. Read et al. (2011) suggest to use the method by Tsoumakas and Katakis (2007) to calculate a threshold value as a lower bound

for the number of chains that need to be in agreement. In more detail, they use the label cardinality L_{Card} which is the average number of labels L per document over all categories (see Equation 3), whereas $L_{i,j}$ is the label assigned to the i-th document for the j-th category.

$$LCARD = \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^{|L|} L_{i,j} \quad (3)$$

The number of the voting classifier chains v that minimizes the difference between the L_{Card} of the training set and the L_{Card} of the test set is considered to be the optimal number of voters \hat{v} which have to be in agreement (see Equation 4).

$$\hat{v} = \arg \min_v |L_{Card}^{train} - L_{Card}^{test}| \quad (4)$$

5 Experimental Results

The models used in the study were trained on the training set containing 14,548 blurbs. For the evaluation the provided validation set containing 2079 blurbs and the Python script was used (Aly et al., 2019). The models were named according to which method or feature was used. An overview of the abbreviations is given in Table 3. Each name thus consist of the SVM implementation used (L or C), the vectorization method (G or F) and whether or not the author information was used as an additional feature (A or nothing).

method/feature	value	abbrev.
SVM	LibSVM / Caret	L / C
vectorization	GloVe / FastText	G / F
author	yes/no	A

Table 3: Explanation of abbreviations for naming the models.

5.1 Comparison of different Models

Overall, five different models were compared with each other, whereas for each model 5 chains were used to prevent anomalies of single chains. The results can be found in Table 4. It can be seen that the CGA-model has the best performance with an F-score of 0.8291, as well as the best results for precision and recall. Furthermore, it can be noted that both models using FastText perform worse than the models using GloVe. The reason could

be the vectorization method itself, yet, it should also be considered that for GloVe pre-trained vectors were used whereas FastText was trained on the blurbs, a much smaller corpus.

Additionally, it can be seen that those models containing the author information as a feature perform better than those not considering this information. This can be easily explained. Most authors wrote books in only one of the eight categories and correspondingly, about 93% of all books only belong to a single category. Hence, using the additional information can improve the results.

Model	Precision	Recall	F1
CFA	0.7758	0.7619	0.7688
CF	0.6515	0.6596	0.6555
CGA	0.8429	0.8157	0.8291
CG	0.7656	0.7794	0.7724
LG	0.7690	0.7955	0.7820

Table 4: Evaluation results for different models each containing five chains.

5.2 Influence of the number of chains

The influence of the number of chains in an ensemble classifier chain was analysed using the example of the CG-model. The results are shown in Table 5. It can be seen that the performance only slightly increases with the number of chains and that the best F1-score is reached with 10 chains. However, the difference to the F1-score of one chain is only minimal. The results confirm the assumption that there is no clear interdependency between the categories at the upper level.

# chains	Precision	Recall	F1
1	0.8083	0.7395	0.7724
2	0.7756	0.7704	0.7730
3	0.7793	0.7614	0.7702
4	0.7743	0.7722	0.7732
5	0.7656	0.7794	0.7724
6	0.7748	0.7776	0.7762
7	0.7748	0.7776	0.7762
8	0.7796	0.7709	0.7752
9	0.7778	0.7785	0.7781
10	0.7802	0.7897	0.7849

Table 5: Performance of the CG model with different numbers of chains.

Moreover, when a single chain is used, the difference between precision and recall is greater than when more chains are used. This can be explained with the threshold value. That means, the more chains need to be in agreement, the more decreases the probability to assign a certain category erroneously, but the more decreases the probability to assign that category generally. In other words, the classifier chain is more conservative in assigning a label. As a consequence less categories are predicted which leads to a higher precision but a lower recall.

5.3 Combination of different models

As was discussed in Section 5.1, the models differ in their performance depending on what features are used. Hence, one idea was to combine the two best performing models (CGA and LG) in order to get better results. Thus, the models were combined by using different numbers of chains from each model. However, no further improvement could be noticed. The best F1-scores are still reached when only the CGA model is used. This indicates that both models misclassify approximately the same books. Consequently, these books cannot be classified correctly, even when combining two models.

5.4 Performance of the different categories

In this section the performance of each category is presented using the example of the CGA-model containing 10 chains and using a minimal consensus of four voters. This model is the one with the best overall performance, reaching an F1-score of 0.841.

We found out that the categories (1, 2, 4, 8, 16, 32, 64, 128) perform very good if they occur as the only category for a book (single label). The category with the best results is category one (*Literatur & Unterhaltung*) with an F-score of 0.931, whereas the category with the worst results is 16 (*Ganzheitliches Bewusstsein*) with an F-score of 0.625. The first category is also the category with the most books in the corpus; about 50% of all books. Thus, it is not surprising that this category has the best classification results. It is also the reason for the good overall performance of the model. In contrast, the performance is much worse when categories are combined (multi label). Then all F-scores are below 0.3333 and most of them are even 0. In all those cases precision as well as recall are very low. Nevertheless, it does not affect the over-

all performance much because it is evaluated using the micro average F-score and the multi label classes only rarely appear (about 7%).

6 Conclusion

This paper deals with multi-label classification of blurbs using ensembles of classifier chains. The best result was achieved with an ensemble of 10 classifier chains, whereas for each classifier a Support Vector Machine with a radial kernel function was used as well as a global vector representation combined with the author information as one feature with an micro F1-score of 0.841. It was shown that neither the number of chains nor the combination of different models had an important influence on the performance. Furthermore, the performance of each category was analysed and the results show that the best performing category is *Literatur und Unterhaltung*, whereas the worst on is *Ganzheitliches Bewusstsein*.

References

- Rami Aly, Steffen Remus, and Chris Biemann. 2019. [Codalab’s evaluation script for germeval-2019 task 1: Shared task on hierarchical classification of blurbs](#). Language Technology Lab, Universität Hamburg.
- Chih-Chung Chang and Chih-Jen Lin. 2001. *LIBSVM: a library for support vector machines*. Department of Computer ScienceNational Taiwan University, Taipei, Taiwan.
- Ashok Chilakapati. 2018. [Word Embeddings and Document Vectors: Part 1. Similarity](#). <http://xplordat.com/2018/09/27/word-embeddings-and-document-vectors-part-1-similarity/>. Accessed 11.06.2019.
- Shantanu Godbole and Sunita Sarawagi. 2004. Discriminative methods for multi-labeled classification. In *Advances in Knowledge Discovery and Data Mining*, pages 22–30, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. 2016. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 2, pages 427–431.
- Max Kuhn. 2019. *Package ‘caret’ Version 6.0-84*.

- Lam Hong Lee, Chin Heng Wan, Rajprasad Rajkumar, and Dino Isa. 2011. [An enhanced support vector machine classification framework by using euclidean distance function for text document categorization](#). *Applied Intelligence*, 37(1):80–99.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. [Efficient estimation of word representations in vector space](#). *Computing Research Repository*, arXiv:1301.3781. Version 3.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. [Distributed representations of words and phrases and their compositionality](#). In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- Malte Pietsch, Timo Möller, and Milos Rusic. 2018. [Pretrained German Word Embeddings](#). deepset GmbH, <https://deepset.ai/german-word-embeddings>. GloVe of german Wikipedia Corpus, accessed 25.05.2019.
- Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2009. [Classifier chains for multi-label classification](#). In *Machine Learning and Knowledge Discovery in Databases*, pages 254–269. Springer Berlin Heidelberg.
- Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. [Classifier chains for multi-label classification](#). *Machine Learning*, 85(3):333–359.
- Steffen Remus, Rami Aly, and Chris Biemann. 2019. Germeval-2019 task 1: Shared task on hierarchical classification of blurbs. In *Proceedings of the GermEval 2019 Workshop*.
- Gerard Salton, Alice Wong, and Chung-Shu Yang. 1975. [A vector space model for automatic indexing](#). *Communications of the ACM*, 18(11):613–620.
- Helmut Schmid. 1994. *TreeTagger - a part-of-speech tagger for many languages*. Institute for Computational Linguistics of the University of Stuttgart.
- Grigoris Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13.
- Yanping Yin and Zhong Jin. 2015. [Document sentiment classification based on the word embedding](#). In *Proceedings of the 4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering 2015*, pages 456–461. Atlantis Press.
- Min-Ling Zhang and Zhi-Hua Zhou. 2005. [A k-nearest neighbor based algorithm for multi-label classification](#). In *IEEE International Conference on Granular Computing*, volume 2, pages 718 – 721.

Multi-Label Multi-Class Hierarchical Classification using Convolutional Seq2Seq

Venkatesh Umaashankar

Ericsson Research / Chennai

venkatesh.u@ericsson.com

Girish Shanmugam S

Intern, Ericsson Research / Chennai

s.girishshanmugam@gmail.com

Abstract

In this paper, We describe our approach for Germeval 2019 Task 1, a hierarchical multi-label multi-class text classification task. This task involves two subtasks where short descriptive text about German books need to be classified into one or multiple (**a**) top level categories (8 classes). (**b**) specific categories (343 classes). We present a novel approach of using Convolutional Seq2Seq modeling for solving both the tasks with a single model. In addition, We use category based random over sampling to handle the imbalance. Our approach reaches f1-micro score of **0.867** on *Subtask (a)* and **0.6722** on *Subtask (b)*. Our approach achieved first rank in *Subtask (a)* and second rank in *Subtask (b)* in the test phase of the shared task. Our code is available in the link <https://gitlab.com/vumaasha/germeval>.

1 Introduction

Multi-label Multi-class Hierarchical classification (MLMCHC) refers to a setting where We can assign one or more labels to each instance (multi-label) where each label can have more than two possible classes (multi-class) that could be organized in a hierarchical structure (hierarchical). MLMCHC problems are common in domains like text classification (Rousu et al. (2006)), image classification (Hsu et al. (2009)) and bioinformatics (Barutcuoglu et al. (2006), Feng et al. (2017)). It is more commonly used in the field of Natural Language Processing (NLP) to classify text documents where a document can have multiple topics associated with them. Unlike the traditional flat classification approach, in MLMCHC the label cardinality (Charte et al., 2015) and number of labels is typically high. Also, the labels are interdependent and their distribution is skewed.

Traditionally, the hierarchical classification problem is solved by a binary relevance approach

where the task is reduced to a flat classification problem by ignoring the label hierarchy and learning an independent binary classifier for each label in the taxonomy or ontology (Tsoumakas et al. (2009)). However, this approach neglects the correlations between labels. Cerri et al. (2016) follow a top-down strategy using neural networks where they use the previous level along with the feature vectors to predict the current level. The issue in this strategy is that the error in a level gets propagated to all the levels following it. Classifier chains (CC) proposed by Read et al. (2011) uses a chain of binary classification problems to model the correlations between labels. This approach is computationally expensive since it relies on training a cascade of classifiers.

Seq2Seq models have achieved tremendous success in machine translation (Bahdanau et al. (2014), Cho et al. (2014)). Li et al. (2018) and Hiramatsu and Wakabayashi (2018) have used RNN in their Seq2Seq models for product taxonomy classification. For machine translation tasks, RNNs are most preferred choice than CNN because of their superior performance on text applications. Gehring et al. (2017) have proposed a Convolutional Seq2Seq model which achieves state-of-the-art accuracy at nine times the speed of recurrent neural systems.

In our approach, we use ***Convolutional Seq2Seq architecture*** to model MLMCHC as a translation task, apply it to Germeval Task1 and evaluate the results. Experiments show that our approach can classify the books more precisely and our model reaches the f1-micro scores of **0.867** on *Subtask (a)* and **0.6722** on *Subtask (b)*. Our approach achieved first rank in *Subtask (a)* and second rank in *Subtask (b)* in the test phase of the shared task. The rest of the paper is organized as follows. We describe the characteristics of the dataset in Section. 2. In Section. 3, we present

our modeling pipeline that explains the sequence of steps in our approach. Feature engineering, imbalance handling and model architecture are explained in Section 3.1, Section 3.3 and Section 3.6 respectively. In Section 3.9 and Section 4 we provide our experiment setup. Finally, in Section 5 we conclude and provide details about the possible future works.

2 Germeval 2019 Task 1 Dataset

The dataset contains the attributes URL, ISBN, title, authors, blurbs, categories, and date of publication corresponding to German books which are crawled from randomhouse.de. The categories could be organized as a hierarchy tree and the metadata corresponding to the hierarchy is provided. This dataset follows the policies described in the RCV1 dataset by Lewis et al. (2004).

343 unique categories are hierarchically structured (8, 93 and 242 on level 1, 2 and 3 respectively). One or more specific categories are assigned to each book. Specific categories need not have to be a leaf node. For instance, the most specific category of a book could be *Romane & Erzählungen*, although *Roman & Erzählungen* has further children categories, such as Romanbiographien.

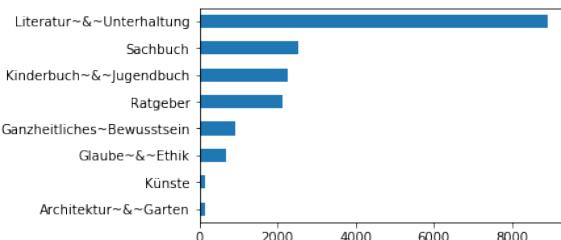


Figure 1: Top Level label distribution

The category distribution for top levels, all levels and specific categories are shown in figures 1, 3 and 2. From the distributions, it can be observed that label distributions on the top level and all levels are much skewed than that of specific categories.

Charte et al. (2015) provides various metrics for characterizing imbalance in Multi-Label Datasets (MLD). According to them, any MLD with a *Max Imbalance Ratio per Label (MeanIR)* value higher than 1.5 (50% more of samples with majority label vs minority label, in average) and *Coefficient of variation of IRLbl (CVIR)* value above 0.2 (20% of variance in the IRLbl values) should be considered

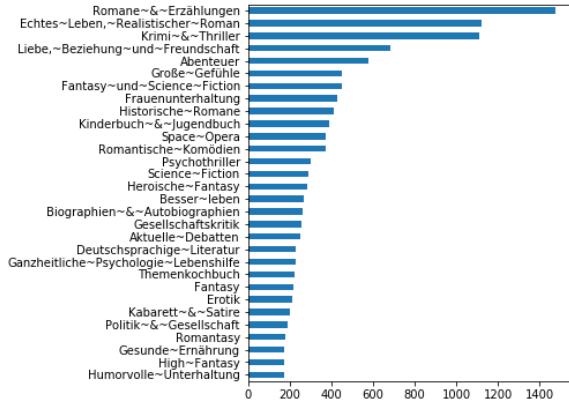


Figure 2: Label distribution for specific categories excluding ancestors for top 30 classes

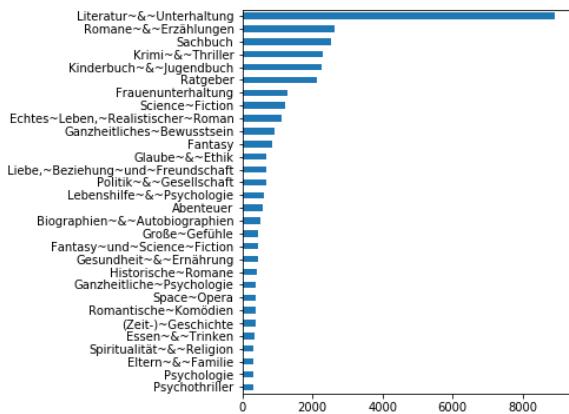


Figure 3: Label distribution including ancestors for top 30 classes

as imbalanced. Our dataset has a high *MeanIR* of **118.46** and high *CVIR* of **1.76** for specific category assignments. This shows that this dataset suffers from severe imbalance.

3 Modeling Pipeline

Our modeling work flow is shown in the figure 4, we used a single model for both the subtasks. Our model was trained to perform the **Subtask (b)**. The results from the **Subtask (b)** are used to generate the results for **Subtask (a)**.

3.1 Data Preparation

We created a hierarchy object from the relationship information provided in the file *hierarchy.txt*. This hierarchy object provides a programmatic interface to get information about any category in the hierarchy. The raw data is provided in the **XML** format. We parsed the XML files using *BeautifulSoup*, a Python package for parsing HTML and XML documents and converted them

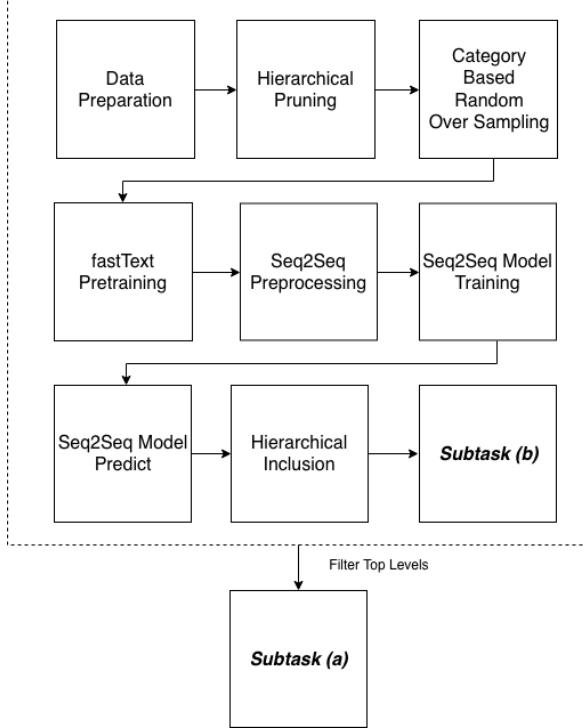


Figure 4: Modeling Pipeline

into **CSV** format.

We concatenated **author**, **title**, **year** along with the blurb into a single text for every book. For the author, title and year, we applied contextual concatenation by inserting markers at the start and end of the context. We concatenated the author information as (@AUTHOR <Author names> AUTHOR@) and similarly for title and year, we used (@TITLE <Title> TITLE@) and (@YEAR <Year> YEAR@). A 13 digit **ISBN** consists of five parts. We extracted the 2 and 3rd part corresponding to **Group** and **Publisher** and appended them to the input text as (@ISBN_GRP <Group> ISBN_GRP@ and @ISBN_PUB <Publisher> ISBN_PUB@). This contextual concatenation allows us to use a single shared embedding representation for multiple modalities such as **author**, **title**, **year**, **ISBN** without losing the context of individual attributes. Also, we preserve the punctuation and special characters by making them valid tokens. We split the labeled data into two splits for training (95%) and validation (5%) and use the unlabelled data as the test split.

3.2 Hierarchical Pruning

For each book, we only pick their specific categories (category tags that contain **label="True"** attribute) as labels. When a book has multiple spe-

cific categories, our label is a concatenated string of all the corresponding specific categories. After predicting these specific categories, the hierarchy object explained in Section. 3.1 can be used to query the ancestor categories of a predicted node, so we avoided including ancestor categories in our labels.

3.3 Category Based Random Over Sampling

In our approach, we only use specific category assignments as labels and skip the corresponding ancestors that can be looked using the hierarchy. Still, as we highlighted already in Section. 2 the training split suffers from severe imbalance. We alleviate this problem by performing category based random oversampling on the training split. Our oversampling algorithm is shown in Algorithm. 1. We oversample the training split by 15%, by using a value of *fraction* = 0.15. We observe improvements in the imbalance metrics, particularly the *MeanIR* reduces to 45.17 from 118.46. The imbalance metrics for specific categories on oversampled data is shown in Table 1 and a comparison of the distribution of top 30 minority classes are shown in the figure 5.

	Actual	Oversampled
Label Cardinality	1.46	1.55
Label Density	0.0043	0.0045
MeanIR	118.46	45.17
MaxIR	1474.00	1474.00
CVIR	1.76	1.85

Table 1: Imbalance characteristics for specific categories in Actual and Oversampled data

```

1 input : dataset, fraction
2 output: oversampled_dataset
3 oversample_size ← size_of(dataset) * fraction;
4 category_wise_freq ← category_frequencies(dataset);
5 category_freq_mean ← mean(category_wise_freq);
6 minority_categories ← {};
7 foreach category, freq ∈ category_wise_freq do
8     if freq < category_freq_mean then
9         minority_categories ← minority_categories ∪ {category};
10    end
11 // Average number of samples to be added for each
12 // minority category
12 mean_increment ← oversample_size/size_of(minority_categories);
13 over_samples ← {};
14 foreach category ∈ minority_categories do
15     mean_diff ← category_freq_mean - frequency(category);
16     samples_to_add ← min(mean_diff, mean_increment);
17     over_samples
18         ← over_samples ∪ random_sample(category, samples_to_add);
19 end
20 oversampled_dataset ← dataset ∪ over_samples

```

Algorithm 1: Category Based ROS

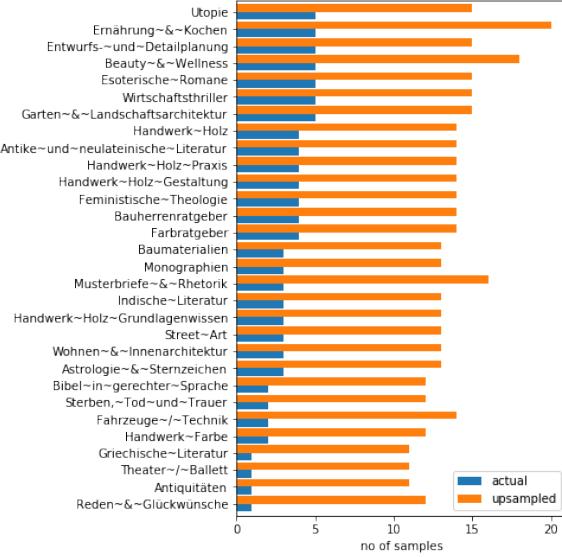


Figure 5: Top 30 Minority categories distribution

3.4 fastText Pretraining

In this phase, We use all the available data (oversampled training, validation and test splits) to learn fastText embeddings for all the tokens in the corpus. fastText provides an implementation for learning character n -grams based continuous word representations proposed by Bojanowski et al. (2017). Each word is considered as a bag of character n -grams and the representation for a word is obtained by the sum of the corresponding character n -gram embeddings. This approach has the advantage of taking morphological features of a word into consideration and also provides representations for words that are not seen in the training corpus.

We learn 2 sets of 100 dimensional embeddings. One for the tokens in the input text (blurbs, title, author, year and ISBN) and another for the categories. Each category in the hierarchy is considered as an individual token. We have a total of 142624 tokens in the input text and 343 tokens in the categories. We use skip-gram and negative sampling options available in fastText. We trained the word embeddings in the input text for 20 epochs and the category embeddings for 100 epochs with a learning rate of 0.05. We use these generic word embeddings to initialize the embeddings in our Seq2Seq model.

3.5 Seq2Seq Preprocessing

We use FAIRSEQ (Ott et al., 2019), a sequence modeling toolkit based on PyTorch. FAIRSEQ

provides predefined architectures and components for Seq2Seq modeling. During preprocessing, FAIRSEQ uses our predefined vocabulary (a global dictionary of tokens from the whole corpus) and encodes the training, validation and test data into integers. The encoded data is saved data in a binary format that supports indexed access and faster loading time.

3.6 Seq2Seq Model Training

A Seq2Seq model mainly contains 2 components, namely *encoder* and *decoder*. The *encoder* converts the input sequence into a fixed size thought vector and the *decoder* sequentially generates the output sequence one step at a time by conditioning on the encoder output and predicted value in the previous time step. Recurrent Neural Networks (RNN) are a popular choice for solving Seq2Seq problems. However, their sequential nature implies that they take longer to train since the training cannot be parallelized.

Gehring et al. (2017) introduced a Seq2Seq architecture that is entirely based on Convolutional Neural Networks (CNN). Convolutional architecture reduces the training time significantly by allowing parallel computation across time and samples. The predictions have to be still performed sequentially, one step at a time. In this architecture, both *encoder* and *decoder* are made of convolutional blocks (figure 6). Each block contains one dimensional convolution with a kernel width k , which is followed by a *Gated Linear Unit* (GLU) (Dauphin et al., 2017) as non-linearity. The GLU facilitates the gradient propagation by implementing a simple gating mechanism over the convolution output. The GLU operation is given by the equation 1, where $Y = [A \ B] \in \mathbb{R}^{2d}$ is the convolution output $A, B, GLU([A \ B]) \in \mathbb{R}^d$, \oplus is point-wise multiplication and σ is sigmoid operation.

$$GLU([A \ B]) = A \oplus \sigma(B) \quad (1)$$

The convolutional blocks are stacked in multiple layers with residual connections from the input of the block to the output of the block to facilitate the flow of gradients during backpropagation. When the input and the output dimensions of block differ, linear projections are used in the residual connections to match the number of dimensions. Multi-layer CNN networks create hierarchical representations over the input sequence.

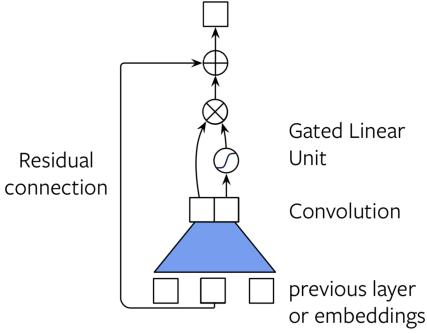


Figure 6: Convolutional Block

This provides a quicker way to obtain dependencies between elements which are far apart in the sequence. With only $\mathcal{O}(\frac{n}{k})$ convolutional operations, the representations for n words in a sequence can be obtained, k is the kernel width. However, in a RNN, it would take $\mathcal{O}(n)$ linear operations to get the representation for n words. Further, an attention network is added to every layer in the decoder.

In our approach, We have modeled the MLMCHC as a Seq2Seq based translation task. We built a model that translates the given input text into a list of categories. Conceptually, this is similar to the Classifier Chaining since at each time step, We model the distribution $P(\text{next category}|\text{previous categories, input text})$. However, our approach does not suffer from the problem of learning several classifiers.

3.6.1 Encoder

Our *encoder* (figure 7) starts with a linear layer of size (100×100) followed by 5 convolution blocks with output sizes $(100, 100, 200, 200, 300)$ and ends with a linear layer of size (300×100) . All the convolutional blocks have a kernel width of 3. We experimented with different number of convolution blocks such 20,15,10,7 and 5. We choose to use 5 convolution blocks finally, since adding more number of blocks did not improve the validation f-score but increased the model complexity and size. The inputs to the encoder are the sum of the word and positional embeddings. Positional embeddings capture the ordering information by embedding the absolute position of the token in the input sequence.

3.6.2 Decoder

Our *decoder* architecture is similar to that of *encoder*. It starts with a linear layer of size $(100 \times$

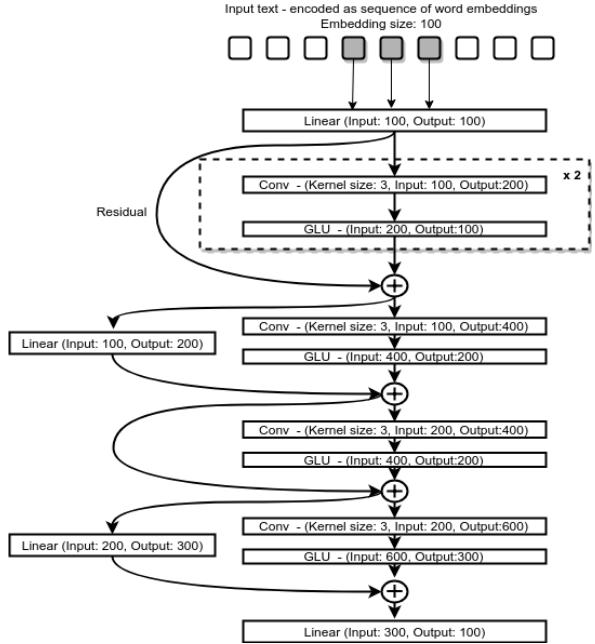


Figure 7: Encoder

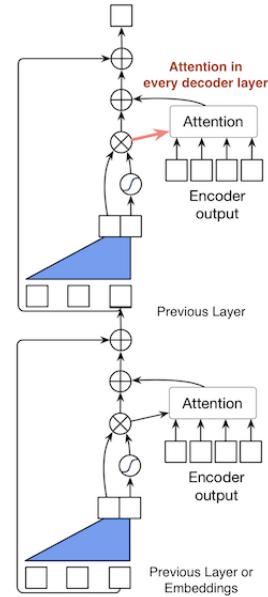


Figure 8: Decoder with Multi-Hop Attention

$100)$ followed by 3 convolutions blocks with output sizes $(100, 100, 200)$, a linear layer of size (200×256) and ends with a linear layer of size (256×343) where 343 is the total number of categories in the hierarchy. Additionally, every *decoder* layer has an attention layer which allows the network to take repeated glimpses at the sequence and decide which input words are more relevant to predict the next word. At every decoder step, a decoder summary is calculated by combining the current decoder state with an embedding of the

previous target element.

3.6.3 Multi-Hop Attention

We use multi-hop attention mechanism similar to Sukhbaatar et al. (2015) by carrying out this process for each step or hops. The attention for each source element is a dot product of the decoder summary and the output of the last encoder block. In multi-hop attention, the attention outputs for a layer is calculated based on the previous layer’s attention results. Hence the decoder has access to attention values of all the previous layers which it uses to predict current layers output.

3.6.4 Model Parameters and Optimization

We initialize the word and category embeddings in the *encoder* and *decoder* using the fastText embeddings that We explained in the Section. 3.4 and fine-tune them as a part of training the Seq2Seq Model. We use Nesterov’s accelerated gradient method with a momentum value of 0.99. We renormalize gradients if their norm exceeds 0.1. We use a fixed learning rate of 0.25. The hyperparameters were chosen based on manual search. We trained the model for 13 epochs, after which the validation loss stopped improving. Our model has a total of 15962496 parameters and the size of our trained model is 122 MB.

3.7 Seq2Seq Model Predict

We use fairseq-generate to generate predictions using the Convolutional Seq2Seq model that We trained in the previous section. The predicted output is a sequence of all specific categories for the given input data. During the prediction phase, We use **beam search** with a beamwidth of 5 to identify the most probable output sequence.

3.8 Hierarchical Inclusion

We use hierarchy object introduced in the Section. 3.1 to query the ancestor categories of specific categories predicted by the Seq2Seq Model. This gives the solution for **Subtask (b)**. We derive the solution for **Subtask (a)** from the solution for **Subtask (b)** by picking the corresponding top levels for predicted specific categories.

3.9 Experiment Setup

(1) Nvidia GPU GEFORCE GTX 1080 Ti 11GB RAM (2) Intel® Xeon® Processor E5-2650 v4 30M Cache, 2.20 GHz, 12 Cores, 24 Threads (3) 250 GB RAM (4) CentOS 7

4 Results

The test evaluation metrics are given in the Table.

2. In the test evaluation of the competition, Our model secured the first rank in **Subtask (a)** with a f1 score of **0.867** and second in **Subtask (b)** with a f1 score of **0.6722**.

	Subtask (a)	Subtask (b)
Precision	0.8923	0.7377
Recall	0.8432	0.6174
F1-Score (micro)	0.867	0.6722
Accuracy	0.8364	0.3791

Table 2: Evaluation metrics on test data

5 Conclusion

In our solution, We have successfully demonstrated that Convolutional Seq2Seq modeling is a promising approach to address MLMCHC. We observed that the oversampling and pretraining phases were key ingredients of our successful recipe. In general, this emphasizes the importance of transfer learning in NLP problems. In the future, We plan to extend our approach by using sophisticated transformers based architecture for both pretraining and modeling phases.

6 Credits

The authors would like to thank their colleagues at Ericsson Research for their support and valuable inputs during this competition. Also, We would like to thank the Facebook research community for developing and maintaining the fastText and FAIRSEQ. Finally, We thank the GermEval competition organizers for fostering a friendly and collaborative environment around this dataset and for answering our questions throughout the competition.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Zafer Barutcuoglu, Robert E Schapire, and Olga G Troyanskaya. 2006. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with

- subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Ricardo Cerri, Rodrigo C Barros, André CPLF de Carvalho, and Yaochu Jin. 2016. Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC bioinformatics*, 17(1):373.
- Francisco Charte, Antonio J Rivera, María J del Jesus, and Francisco Herrera. 2015. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, 163:3–16.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
- Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR.org.
- Shou Feng, Ping Fu, and Wenbin Zheng. 2017. A hierarchical multi-label classification algorithm for gene function prediction. *Algorithms*, 10(4):138.
- Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR.org.
- Makoto Hiramatsu and Kei Wakabayashi. 2018. Encoder-decoder neural networks for taxonomy classification.
- Daniel J Hsu, Sham M Kakade, John Langford, and Tong Zhang. 2009. Multi-label prediction via compressed sensing. In *Advances in neural information processing systems*, pages 772–780.
- David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.
- Maggie Yundi Li, Liling Tan, Stanley Kok, and Ewa Szymanska. 2018. Unconstrained product categorization with sequence-to-sequence models.
- Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.
- Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine learning*, 85(3):333.
- Juho Rousu, Craig Saunders, Sandor Szemek, and John Shawe-Taylor. 2006. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7(Jul):1601–1626.
- Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.
- Grigoris Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2009. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer.

Enriching BERT with Knowledge Graph Embeddings for Document Classification

Malte Ostendorff^{1,2}, Peter Bourgonje¹, Maria Berger¹, Julián Moreno-Schneider¹, Georg Rehm¹, Bela Gipp²

¹Speech and Language Technology, DFKI GmbH, Germany

first.last@dfki.de

²University of Konstanz, Germany

first.last@uni-konstanz.de

Abstract

In this paper, we focus on the classification of books using short descriptive texts (cover blurbs) and additional metadata. Building upon BERT, a deep neural language model, we demonstrate how to combine text representations with metadata and knowledge graph embeddings, which encode author information. Compared to the standard BERT approach we achieve considerably better results for the classification task. For a more coarse-grained classification using eight labels we achieve an F1-score of 87.20, while a detailed classification using 343 labels yields an F1-score of 64.70. We make the source code and trained models of our experiments publicly available.

1 Introduction

With ever-increasing amounts of data available, there is an increase in the need to offer tooling to speed up processing, and eventually making sense of this data. Because fully-automated tools to extract meaning from any given input to any desired level of detail have yet to be developed, this task is still at least supervised, and often (partially) resolved by humans; we refer to these humans as knowledge workers. Knowledge workers are professionals that have to go through large amounts of data and consolidate, prepare and process it on a daily basis. This data can originate from highly diverse portals and resources and depending on type or category, the data needs to be channelled through specific down-stream processing pipelines. We aim to create a platform for *curation technologies* that can deal with such data from diverse sources and that provides natural language processing (NLP) pipelines tailored to particular content types and genres, rendering this initial classification an important sub-task.

In this paper, we work with the dataset of the 2019 GermEval shared task on hierarchical text

classification (Remus et al., 2019) and use the pre-defined set of labels to evaluate our approach to this classification task¹.

Deep neural language models have recently evolved to a successful method for representing text. In particular, Bidirectional Encoder Representations from Transformers (BERT; Devlin et al., 2019) outperformed previous state-of-the-art methods by a large margin on various NLP tasks. We adopt BERT for text-based classification and extend the model with additional metadata provided in the context of the shared task, such as author, publisher, publishing date, etc.

A key contribution of this paper is the inclusion of additional (meta) data using a state-of-the-art approach for text processing. Being a transfer learning approach, it facilitates the task solution with external knowledge for a setup in which relatively little training data is available. More precisely, we enrich BERT, as our pre-trained text representation model, with knowledge graph embeddings that are based on Wikidata (Vrandecic and Krötzsch, 2014), add metadata provided by the shared task organisers (title, author(s), publishing date, etc.) and collect additional information on authors for this particular document classification task. As we do not rely on text-based features alone but also utilize document metadata, we consider this as a document classification problem. The proposed approach is an attempt to solve this problem exemplary for single dataset provided by the organisers of the shared task.

2 Related Work

A central challenge in work on genre classification is the definition of a both rigid (for theoretical purposes) and flexible (for practical purposes) mode

¹<https://www.inf.uni-hamburg.de/en/inst/ab/lt/resources/data/germeval-2019-hmc.html>

of representation that is able to model various dimensions and characteristics of arbitrary text genres. The size of the challenge can be illustrated by the observation that there is no clear agreement among researchers regarding actual genre labels or their scope and consistency. There is a substantial amount of previous work on the definition of genre taxonomies, genre ontologies, or sets of labels (Biber, 1988; Lee, 2002; Sharoff, 2018; Underwood, 2014; Rehm, 2005). Since we work with the dataset provided by the organisers of the 2019 GermEval shared task, we adopt their hierarchy of labels as our genre palette. In the following, we focus on related work more relevant to our contribution.

With regard to **text and document classification**, BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) is a pre-trained embedding model that yields state of the art results in a wide span of NLP tasks, such as question answering, textual entailment and natural language inference learning (Artetxe and Schwenk, 2018). Adhikari et al. (2019) are among the first to apply BERT to document classification. Acknowledging challenges like incorporating syntactic information, or predicting multiple labels, they describe how they adapt BERT for the document classification task. In general, they introduce a fully-connected layer over the final hidden state that contains one neuron each representing an input token, and further optimize the model choosing soft-max classifier parameters to weight the hidden state layer. They report state of the art results in experiments based on four popular datasets. An approach exploiting Hierarchical Attention Networks is presented by Yang et al. (2016). Their model introduces a hierarchical structure to represent the hierarchical nature of a document. Yang et al. (2016) derive attention on the word and sentence level, which makes the attention mechanisms react flexibly to long and short distant context information during the building of the document representations. They test their approach on six large scale text classification problems and outperform previous methods substantially by increasing accuracy by about 3 to 4 percentage points. Aly et al. (2019) (the organisers of the GermEval 2019 shared task on hierarchical text classification) use shallow capsule networks, reporting that these work well on structured data for example in the field of visual inference, and

outperform CNNs, LSTMs and SVMs in this area. They use the Web of Science (WOS) dataset and introduce a new real-world scenario dataset called Blurb Genre Collection (BGC)².

With regard to **external resources to enrich the classification task**, Zhang et al. (2019) experiment with external knowledge graphs to enrich embedding information in order to ultimately improve language understanding. They use structural knowledge represented by Wikidata entities and their relation to each other. A mix of large-scale textual corpora and knowledge graphs is used to further train language representation exploiting ERNIE (Sun et al., 2019), considering lexical, syntactic, and structural information. Wang et al. (2009) propose and evaluate an approach to improve text classification with knowledge from Wikipedia. Based on a bag of words approach, they derive a thesaurus of concepts from Wikipedia and use it for document expansion. The resulting document representation improves the performance of an SVM classifier for predicting text categories.

3 Dataset and Task

Our experiments are modelled on the GermEval 2019 shared task and deal with the classification of books. The dataset contains 20,784 German books. Each record has:

- A title.
- A list of authors. The average number of authors per book is 1.13, with most books (14,970) having a single author and one outlier with 28 authors.
- A short descriptive text (**blurb**) with an average length of 95 words.
- A URL pointing to a page on the publisher’s website.
- An ISBN number.
- The date of publication.

The books are labeled according to the hierarchy used by the German publisher Random House. This taxonomy includes a mix of genre and topical categories. It has eight top-level genre categories, 93 on the second level and 242 on the

²Note that this is not the dataset used in the shared task.

most detailed third level. The eight top-level labels are ‘Ganzheitliches Bewusstsein’ (*holistic awareness/consciousness*), ‘Künste’ (*arts*), ‘Sachbuch’ (*non-fiction*), ‘Kinderbuch & Jugendbuch’ (*children and young adults*), ‘Ratgeber’ (*counselor/advisor*), ‘Literatur & Unterhaltung’ (*literature and entertainment*), ‘Glaube & Ethik’ (*faith and ethics*), ‘Architektur & Garten’ (*architecture and garden*). We refer to the shared task description³ for details on the lower levels of the ontology.

Note that we do not have access to any of the full texts. Hence, we use the blurbs as input for BERT. Given the relatively short average length of the blurbs, this considerably decreases the amount of data points available for a single book.

The shared task is divided into two sub-task. Sub-task A is to classify a book, using the information provided as explained above, according to the top-level of the taxonomy, selecting one or more of the eight labels. Sub-task B is to classify a book according to the detailed taxonomy, specifying labels on the second and third level of the taxonomy as well (in total 343 labels). This renders both sub-tasks a multi-label classification task.

4 Experiments

As indicated in Section 1, we base our experiments on BERT in order to explore if it can be successfully adopted to the task of book or document classification. We use the pre-trained models and enrich them with additional metadata and tune the models for both classification sub-tasks.

4.1 Metadata Features

In addition to the metadata provided by the organisers of the shared task (see Section 3), we add the following features.

- Number of authors.
- Academic title (Dr. or Prof.), if found in author names (0 or 1).
- Number of words in title.
- Number of words in blurb.
- Length of longest word in blurb.
- Mean word length in blurb.

³<https://competitions.codalab.org/competitions/20139>

	Train	Validation	Test
Gender	12,681 (87%)	1,834 (88%)	3,641 (88%)
Author emb.	10,407 (72%)	1,549 (75%)	3,010 (72%)
Total books	14,548	2,079	4,157

Table 1: Availability of additional data with respect to the dataset (relative numbers in parenthesis).

- Median word length in blurb.
- Age in years after publication date.
- Probability of first author being male or female based on the *Gender-by-Name* dataset⁴. Available for 87% of books in training set (see Table 1).

The statistics (length, average, etc.) regarding blurbs and titles are added in an attempt to make certain characteristics explicit to the classifier. For example, books labeled ‘Kinderbuch & Jugendbuch’ (*children and young adults*) have a title that is on average 5.47 words long, whereas books labeled ‘Künste’ (*arts*) on average have shorter titles of 3.46 words. The binary feature for academic title is based on the assumption that academics are more likely to write non-fiction. The gender feature is included to explore (and potentially exploit) whether or not there is a gender-bias for particular genres.

4.2 Author Embeddings

Whereas one should not judge a book by its cover, we argue that additional information on the author can support the classification task. Authors often adhere to their specific style of writing and are likely to specialize in a specific genre.

To be precise, we want to include author identity information, which can be retrieved by selecting particular properties from, for example, the Wikidata knowledge graph (such as date of birth, nationality, or other biographical features). A drawback of this approach, however, is that one has to manually select and filter those properties that improve classification performance. This is why, instead, we follow a more generic approach and utilize automatically generated graph embeddings as author representations.

⁴Probability of given names being male/female based on US names from 1930-2015. <https://data.world/howarder/gender-by-name>

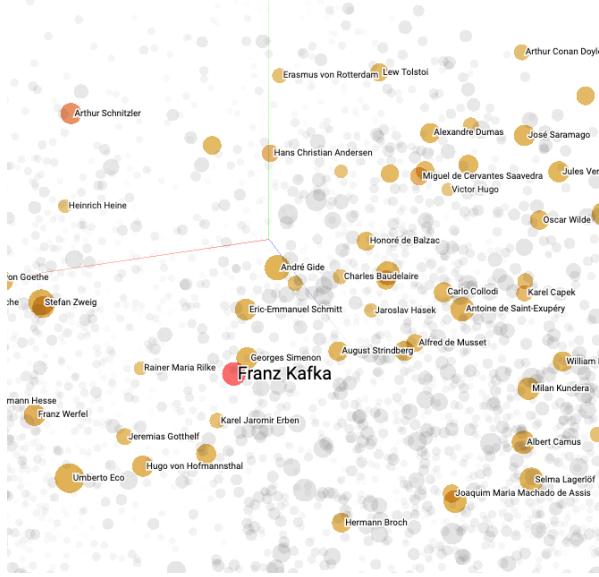


Figure 1: Visualization of Wikidata embeddings for *Franz Kafka* (3D-projection with PCA)⁵. Nearest neighbours in original 200D space: *Arthur Schnitzler*, *E.T.A Hoffmann* and *Hans Christian Andersen*.

Graph embedding methods create dense vector representations for each node such that distances between these vectors predict the occurrence of edges in the graph. The node distance can be interpreted as topical similarity between the corresponding authors.

We rely on pre-trained embeddings based on PyTorch BigGraph (Lerer et al., 2019). The graph model is trained on the full Wikidata graph, using a translation operator to represent relations⁶. Figure 1 visualizes the locality of the author embeddings.

To derive the author embeddings, we look up Wikipedia articles that match with the author names and map the articles to the corresponding Wikidata items⁷. If a book has multiple authors, the embedding of the first author for which an embedding is available is used. Following this method, we are able to retrieve embeddings for 72% of the books in the training and test set (see Table 1).

4.3 Pre-trained German Language Model

Although the pre-trained BERT language models are multilingual and, therefore, support German, we rely on a BERT model that was exclusively

⁶Pre-trained Knowledge Graph Embeddings. <https://github.com/facebookresearch/PyTorch-BigGraph#pre-trained-embeddings>

⁷Mapping Wikipedia pages to Wikidata IDs and vice versa. <https://github.com/jcklie/wikimapper>

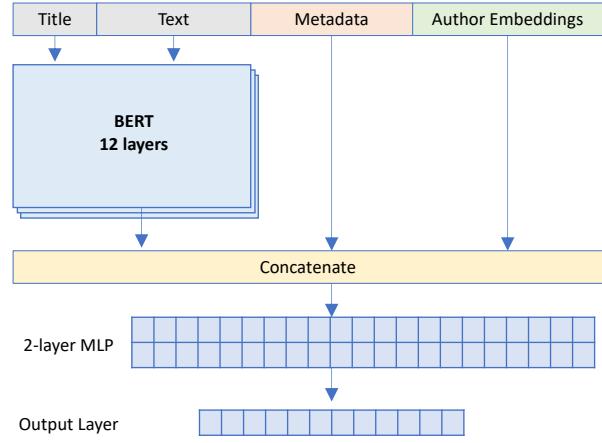


Figure 2: Model architecture used in our experiments. Text-features are fed through BERT, concatenated with metadata and author embeddings and combined in a multilayer perceptron (MLP).

pre-trained on German text, as published by the German company Deepset AI⁸. This model was trained from scratch on the German Wikipedia, news articles and court decisions⁹. Deepset AI reports better performance for the German BERT models compared to the multilingual models on previous German shared tasks (GermEval2018-Fine and GermEval 2014).

4.4 Model Architecture

Our neural network architecture, shown in Figure 2, resembles the original BERT model (Devlin et al., 2019) and combines text- and non-text features with a multilayer perceptron (MLP).

The BERT architecture uses 12 hidden layers, each layer consists of 768 units. To derive contextualized representations from textual features, the book title and blurb are concatenated and then fed through BERT. To minimize the GPU memory consumption, we limit the input length to 300 tokens (which is shorter than BERT’s hard-coded limit of 512 tokens). Only 0.25% of blurbs in the training set consist of more than 300 words, so this cut-off can be expected to have minor impact.

The non-text features are generated in a separate preprocessing step. The metadata features are represented as a ten-dimensional vector (two dimensions for gender, see Section 4.1). Author embedding vectors have a length of 200 (see Sec-

⁸Details on BERT-German training procedure: <https://deepset.ai/german-bert>

⁹German legal documents used to train BERT-German: <http://openlegaldata.io/research/2019/02/19/court-decision-dataset.html>

tion 4.2). In the next step, all three representations are concatenated and passed into a MLP with two layers, 1024 units each and ReLu activation function. During training, the MLP is supposed to learn a non-linear combination of its input representations. Finally, the output layer does the actual classification. In the SoftMax output layer each unit corresponds to a class label. For sub-task A the output dimension is eight. We treat sub-task B as a standard multi-label classification problem, i. e., we neglect any hierarchical information. Accordingly, the output layer for sub-task B has 343 units. When the value of an output unit is above a given threshold the corresponding label is predicted, whereby thresholds are defined separately for each class. The optimum was found by varying the threshold in steps of 0.1 in the interval from 0 to 1.

4.5 Implementation

Training is performed with batch size $b = 16$, dropout probability $d = 0.1$, learning rate $\eta = 2^{-5}$ (Adam optimizer) and 5 training epochs. These hyperparameters are the ones proposed by Devlin et al. (2019) for BERT fine-tuning. We did not experiment with hyperparameter tuning ourselves except for optimizing the classification threshold for each class separately. All experiments are run on a GeForce GTX 1080 Ti (11 GB), whereby a single training epoch takes up to 10min. If there is no single label for which prediction probability is above the classification threshold, the most popular label (*Literatur & Unterhaltung*) is used as prediction.

4.6 Baseline

To compare against a relatively simple baseline, we implemented a Logistic Regression classifier chain from scikit-learn (Pedregosa et al., 2011). This baseline uses the text only and converts it to TF-IDF vectors. As with the BERT model, it performs 8-class multi-label classification for sub-task A and 343-class multi-label classification for sub-task B, ignoring the hierarchical aspect in the labels.

5 Results

Table 2 shows the results of our experiments. As prescribed by the shared task, the essential evaluation metric is the micro-averaged F1-score. All scores reported in this paper are obtained using

models that are trained on the training set and evaluated on the validation set. For the final submission to the shared task competition, the best-scoring setup is used and trained on the training and validation sets combined.

We are able to demonstrate that incorporating metadata features and author embeddings leads to better results for both sub-tasks. With an F1-score of 87.20 for task A and 64.70 for task B, the setup using BERT-German with metadata features and author embeddings (1) outperforms all other setups. Looking at the precision score only, BERT-German with metadata features (2) but without author embeddings performs best.

In comparison to the baseline (7), our evaluation shows that deep transformer models like BERT considerably outperform the classical TF-IDF approach, also when the input is the same (using the title¹⁰ and blurb only). BERT-German (4) and BERT-Multilingual (5) are only using text-based features (title and blurb), whereby the text representations of the BERT-layers are directly fed into the classification layer.

To establish the information gain of author embeddings, we train a linear classifier on author embeddings, using this as the only feature. The author-only model (6) is exclusively evaluated on books for which author embeddings are available, so the numbers are based on a slightly smaller validation set. With an F1-score of 61.99 and 32.13 for sub-tasks A and B, respectively, the author model yields the worst result. However, the information contained in the author embeddings help improve performance, as the results of the best-performing setup show. When evaluating the best model (1) only on books for that author embeddings are available, we find a further improvement with respect to F1 score (task A: from 87.20 to 87.81; task B: 64.70 to 65.74).

6 Discussion

The best performing setup uses BERT-German with metadata features and author embeddings. In this setup the most data is made available to the model, indicating that, perhaps not surprisingly, more data leads to better classification performance. We expect that having access to the actual text of the book will further increase perfor-

¹⁰The baseline model uses the blurbs only, without the title, but we do not expect that including the title in the input would make up for the considerable gap between the two.

Model / Features	Sub-Task A – 8 labels			Sub-Task B – 343 labels		
	F1	Prec.	Recall	F1	Prec.	Recall
(1) BERT-German + Metadata + Author	87.20	88.76	85.70	64.70	83.78	52.70
(2) BERT-German + Metadata	86.90	89.65	84.30	63.96	83.94	51.67
(3) BERT-German + Author	86.84	89.02	84.75	64.41	82.02	53.03
(4) BERT-German	86.65	89.65	83.86	60.51	83.44	47.47
(5) BERT-Base-Multilingual-Cased	83.94	86.31	81.70	54.08	82.63	40.19
(6) Author	61.99	75.59	52.54	32.13	72.39	20.65
(7) Baseline	77.00	79.00	74.00	45.00	67.00	34.00
Results of best model (1) on test set	88.00	85.00	86.00	78.00	52.00	62.00

Table 2: Evaluation scores (micro avg.) on validation set with respect to the features used for classification. The model with BERT-German, metadata and author embeddings yields the highest F1-scores on both tasks and was accordingly submitted to the GermEval 2019 competition. The scores in the last row are the result on the test set as reported by Remus et al., 2019.

Title / Author	Correct Labels	Predicted Labels
<i>Coenzym Q10</i> Dr. med. Gisela Rauch-Petz	Ratgeber (I); Gesundheit & Ernährung (II)	Gesundheit & Ernährung (II)
<i>Gelebte Wertschätzung</i> Barbara von Meibom	Glaube & Ethik (I); Psychologie & Spiritualität (II)	Sachbuch (I); Politik & Gesellschaft (II)
<i>Wie Romane entstehen</i> Hanns-Josef Ortheil, Klaus Siblewski	Literatur & Unterhaltung (I); Sachbuch (I); Romane & Erzählungen (II); Briefe, Essays, Gespräche (II)	Literatur & Unterhaltung (I)
<i>Das Grab ist erst der Anfang</i> Kathy Reichs	Literatur & Unterhaltung (I); Krimi & Thriller (II)	Literatur & Unterhaltung (I); Krimi & Thriller (II)

Table 3: Book examples and their correct and predicted labels. Hierarchical label level is in parenthesis.

mance. The average number of words per blurb is 95 and only 0.25% of books exceed our cut-off point of 300 words per blurb. In addition, the distribution of labeled books is imbalanced, i.e. for many classes only a single digit number of training instances exist (Fig. 3). Thus, this task can be considered a low resource scenario, where including related data (such as author embeddings and author identity features such as gender and academic title) or making certain characteristics more explicit (title and blurb length statistics) helps. Furthermore, it should be noted that the blurbs do not provide summary-like abstracts of the book, but instead act as teasers, intended to persuade the reader to buy the book.

As reflected by the recent popularity of deep transformer models, they considerably outperform the Logistic Regression baseline using TF-IDF representation of the blurbs. However, for the simpler sub-task A, the performance difference between the baseline model and the multilingual BERT model is only six points, while consum-

ing only a fraction of BERT’s computing resources. The BERT model trained for German (from scratch) outperforms the multilingual BERT model by under three points for sub-task A and over six points for sub-task B, confirming the findings reported by the creators of the BERT-German models for earlier GermEval shared tasks.

While generally on par for sub-task A¹¹, for sub-task B there is a relatively large discrepancy between precision and recall scores. In all setups, precision is considerably higher than recall. We expect this to be down to the fact that for some of the 343 labels in sub-task B, there are very few instances. This means that if the classifier predicts a certain label, it is likely to be correct (i. e., high precision), but for many instances having low-frequency labels, this low-frequency label is never predicted (i. e., low recall).

As mentioned in Section 4.4, we neglect the hierarchical nature of the labels and flatten the hierarchy (with a depth of three levels) to a sin-

¹¹Except for the Author-only setup.

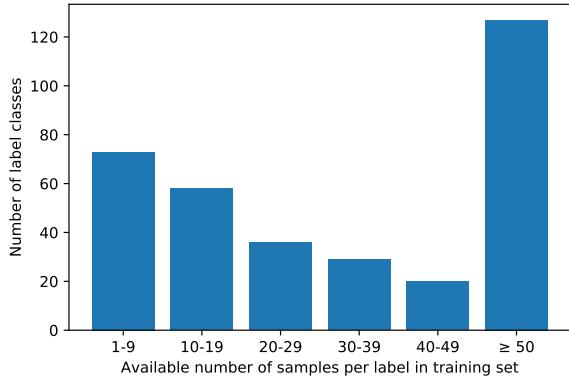


Figure 3: In sub-task B for many low-hierarchical labels only a small number of training samples exist, making it more difficult to predict the correct label.

gle set of 343 labels for sub-task B. We expect this to have negative impact on performance, because it allows a scenario in which, for a particular book, we predict a label from the first level and also a non-matching label from the second level of the hierarchy. The example *Coenzym Q10* (Table 3) demonstrates this issue. While the model correctly predicts the second level label *Gesundheit & Ernährung* (health & diet), it misses the corresponding first level label *Ratgeber* (advisor). Given the model’s tendency to higher precision rather than recall in sub-task B, as a post-processing step we may want to take the most detailed label (on the third level of the hierarchy) to be correct and manually fix the higher level labels accordingly. We leave this for future work and note that we expect this to improve performance, but it is hard to say by how much. We hypothesize that an MLP with more and bigger layers could improve the classification performance. However, this would increase the number of parameters to be trained, and thus requires more training data (such as the book’s text itself, or a summary of it).

7 Conclusions and Future Work

In this paper we presented a way of enriching BERT with knowledge graph embeddings and additional metadata. Exploiting the linked knowledge that underlies Wikidata improves performance for our task of document classification. With this approach we improve the standard BERT models by up to four percentage points in accuracy. Furthermore, our results reveal that with task-specific information such as author names

and publication metadata improves the classification task essentially compared a text-only approach. Especially, when metadata feature engineering is less trivial, adding additional task-specific information from an external knowledge source such as Wikidata can help significantly. The source code of our experiments and the trained models are publicly available¹².

Future work comprises the use of hierarchical information in a post-processing step to refine the classification. Another promising approach to tackle the low resource problem for task B would be to use label embeddings. Many labels are similar and semantically related. The relationships between labels can be utilized to model in a joint embedding space (Augenstein et al., 2018). However, a severe challenge with regard to setting up label embeddings is the quite heterogeneous category system that can often be found in use online. The Random House taxonomy (see above) includes category names, i. e., labels, that relate to several different dimensions including, among others, genre, topic and function.

This work is done in the context of a larger project that develops a platform for curation technologies. Under the umbrella of this project, the classification of pieces of incoming text content according to an ontology is an important step that allows the routing of this content to particular, specialized processing workflows, including parameterising the included pipelines. Depending on content type and genre, it may make sense to apply OCR post-processing (for digitized books from centuries ago), machine translation (for content in languages unknown to the user), information extraction, or other particular and specialized procedures. Constructing such a generic ontology for digital content is a challenging task, and classification performance is heavily dependent on input data (both in shape and amount) and on the nature of the ontology to be used (in the case of this paper, the one predefined by the shared task organisers). In the context of our project, we continue to work towards a maximally generic content ontology, and at the same time towards applied classification architectures such as the one presented in this paper.

¹²<https://ostendorff.org/r/germeval19>

Acknowledgments

This research is funded by the German Federal Ministry of Education and Research (BMBF) through the “Unternehmen Region”, instrument “Wachstumskern” QURATOR (grant no. 03WKDA1A). We would like to thank the anonymous reviewers for comments on an earlier version of this manuscript.

References

- Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: BERT for document classification. *CoRR*, abs/1904.08398.
- Rami Aly, Steffen Remus, and Chris Biemann. 2019. Hierarchical multi-label classification of text with capsule networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 323–330, Florence, Italy. Association for Computational Linguistics.
- Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *CoRR*, abs/1812.10464.
- Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. 2018. Multi-task learning of pairwise sequence classification tasks over disparate label spaces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1896–1906, New Orleans, Louisiana. Association for Computational Linguistics.
- Douglas Biber. 1988. *Variation across Speech and Writing*. Cambridge University Press.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- David Y. W. Lee. 2002. Genres, Registers, Text Types, Domains, and Styles: Clarifying the Concepts and Navigating a Path through the BNC Jungle. *Language Learning and Technology*, 5(3):37–72.
- Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. 2019. PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of the 2nd SysML Conference*, Palo Alto, CA, USA.
- Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Georg Rehm. 2005. *Hypertextsorten: Definition – Struktur – Klassifikation*. Ph.D. thesis, Justus-Liebig-Universität Gießen, Norderstedt.
- Steffen Remus, Rami Aly, and Chris Biemann. 2019. GermEval 2019 Task 1 : Hierarchical Classification of Blurbs. In *Proceedings of the GermEval 2019 Workshop*, pages 1–13, Erlangen, Germany.
- Serge Sharoff. 2018. Functional text dimensions for the annotation of web corpora. *Corpora*, 13(1):65–95.
- Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv:1904.09223*.
- Ted Underwood. 2014. *Understanding Genre in a Collection of a Million Volumes, Interim Report*. figshare.
- Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.
- Pu Patrick Wang, Jingjie Hu, Hua-Jun Zeng, and Zhi-gang Chen. 2009. Using wikipedia knowledge to improve text classification. *Knowledge and Information Systems*, 19(3):265–281.
- Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.
- Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

COMTRAVO-DS team at GermEval 2019 Task 1 on Hierarchical Classification of Blurbs

David S. Batista

Comtravo GmbH

david.batista@comtravo.com

Matti Lyra

Comtravo GmbH

matti.lyra@comtravo.com

Abstract

We present two systems developed by the Comtravo Data Science team for the GermEval’19 Task 1 on hierarchical classification of blurbs. The challenge is a document classification task where the hierarchical structure of each document needs to be captured. Our systems achieved the 13th place out of 19 submissions for Sub-Task A and the 11th place out of 19 submissions for Sub-Task B. We describe in detail these two systems pointing out the advantages and disadvantages of each as well as laying out future research directions.

1 Introduction

This paper describes the approach taken by the Comtravo Data Science team for the GermEval’19 Task 1 ([Remus et al., 2019](#)). The task aimed at developing systems to tackle the task of multi-label hierarchical classification of text.

Several real-world classification problems are naturally cast within a hierarchy, where the labels to be predicted are organized in an hierarchy. Typically the hierarchies form a tree, several trees (a forest) or a directed acyclic graph.

Examples of hierarchical document categorization are for instance categorizing news articles into an hierarchy of categories ([Lewis et al., 2004](#)), a web page into a web directory structure, Wikipedia articles into the Wikipedia taxonomy ([Partalas et al., 2015](#)), or in biomedical literature, for instance, the assignment of Medical Subject Headings to PubMed abstracts ([Lipscomb, 2000](#)).

We developed two distinct approaches, one based on a local classifier strategy, where different classifiers are trained according to the hierarchical structure of the label space, another approach uses a single classifier which tries to naively predict the entire label hierarchy for each sample.

This paper is organized as follows, in Section 2 we describe the task in detail and give a statistical description of the provided dataset. In Section 3 we briefly describe some of the proposed approaches in the literature for hierarchical document classification. In Section 4 we describe our approaches to tackle both sub-tasks. Section 5 details the our experimental setup and results. Finally in Section 6 we outline some ideas for future work.

2 Task

The GermEval 2019 Task 1 on hierarchical classification of blurbs involved the classification of german language books into genres given a book’s blurb i.e., a short textual description of the book and related meta-data. The competition contained two sub-tasks:

- Sub-Task A: classify German books into one or multiple general genres, a non-hierarchical multi-label classification task with a total of 8 classes.
- Sub-Task B: targets hierarchical multi-label classification into multiple writing genres. In addition to the general genres from Sub-Task A, any number of sub-genres of increasing specificity can be assigned to a book.

2.1 Dataset

The dataset made available for these tasks contains 3 label levels organized in a hierarchy; any book can be associated with more than one label at any given level of the hierarchy. In the hierarchy every child-label has exactly one parent-label.

The hierarchy contains a total of 343 distinct classes, 3 datasets were provided: 14 548 samples available for training, 2 079 for development and 4 157 for testing. Tables 1, 2, 3 contain a detailed

Training set	
Avg. length of blurb (tokens)	96.78
Std. deviation σ (tokens)	39.63
Avg. length of blurb (sentences)	6.55
Std. deviation σ (sentences)	2.76
Nr. unique tokens original	114 903
Nr. unique tokens lowercase	107 998
Total number of genres	343
Possible genres per level (1;2;3)	8; 93; 242
Avg. genres per blurb	3.1
Std. deviation σ	1.36
Avg. genres per blurb at level (1;2;3)	1.06; 1.34; 0.69
Std. deviation σ	0.27; 0.76; 0.79
Avg. blurb per co-occurrence	6.48
Co-Occurrence std. deviation	35.90
Nr. samples with leaf nodes at:	
- Level 1	1.9% (311)
- Level 2	44.6% (7.422)
- Level 3	53.5% (8.894)
Total number of samples	14 548

Table 1: Quantitative analysis of the training dataset.

description of the datasets provided i.e.: training, development and test, respectively. One can see that in terms of tokens and sentences the 3 datasets are aligned, and also between training and development in terms of labels per blurb, and labels per blurb per level.

3 Hierarchical Document Classification

There are have been different strategies to approach the problem of hierarchical classifying a document (Silla and Freitas, 2011; Wehrmann et al., 2017; Kowsari et al., 2017). Within the context of GermEval’19 Task 1 we explored two main strategies: a local classifier and a global classifier.

3.1 Local Classifier

The local classifier strategy is one way to approach the hierarchical document classification task and it was first proposed, to the best of our knowledge, in the seminal work of Koller and Sahami (1997), it is also sometimes referred to as top down approach in the literature.

There are different approaches, based on the idea of a local classifier, depending on how they use the local information and devise a strategy to build several classifiers.

3.1.1 A classifier per node

The *local classifier per node approach* consists of training one binary classifier in a one-versus-rest scenario for each node in the hierarchy, where each label in the hierarchy is a node. Normally,

Development set	
Avg. length of blurb (tokens)	98.71
Std. deviation σ (tokens)	46.29
Avg. length of blurb (sentences)	6.68
Std. deviation σ (sentences)	3.80
Nr. unique tokens original	33 599
Nr. unique tokens lowercase	31 818
Total number of genres	343
Possible genres per level (1;2;3)	8; 93; 242
Avg. genres per blurb	3.1
Std. deviation σ	1.39
Avg. genres per blurb at level (1;2;3)	1.07;1.35;0.69
Std. deviation σ	0.27;0.80;0.79
Avg. blurb per co-occurrence	3.08
Co-Occurrence std. deviation	8.19
Nr. samples with leaf nodes at:	
- Level 1	1.6% (34)
- Level 2	44.8% (932)
- Level 3	53.6% (1113)
Total number of samples	2 079

Table 2: Quantitative analysis of the development dataset.

Test set	
Avg. length of blurb (tokens)	96.91
Std. deviation σ (tokens)	39.83
Avg. length of blurb (sentences)	6.55
Std. deviation σ (sentences)	2.62
Total number of samples	4 157

Table 3: Quantitative analysis of the test dataset.

the negative training data is taken from the same level in the label hierarchy as the positive data.

During prediction a top-down strategy is applied, the output of each binary classifier is a prediction of whether or not a given test sample belongs to the classifier’s predicted class. This approach is naturally multi-label since it is possible to predict multiple labels at each level of the hierarchy.

This approach, however, is prone to label inconsistency. Consider a document that has, for the first level, labels 1, 2 and 3, and, for the second level, labels 1.1, 1.2. Since the classifiers for nodes 1 and 1.1 are independently trained, it is possible to classify a sample as having labels 1.2 and 1.1 but not the parent label 1. This approach should, therefore, be complemented by a post-processing method that tries to correct the label inconsistency.

3.1.2 A classifier per parent node

In a *classifier per parent node* approach, a multi-class classifier, possibly also multi-label, is trained for each parent node in the label hierarchy. The classifier is trained to classify the probability of

a given sample belonging to each of the parent's child nodes. In this case, a parent node is every label in the hierarchy-tree which has one or more child labels.

Given a test sample, first the top-level classifier is applied, then for every top-level predicted label (e.g., class 2 and 3) its child classifiers, e.g.: a classifier trained to predict the 2.x labels and another for 3.x labels, and so on until the last level is reached.

Note that the sub-classifiers are only trained with the children of each respective parent label, therefore this approach avoids the label inconsistency problem and respects the constraints of class-membership defined by the label hierarchy.

3.1.3 A classifier per level

The *local classifier per level* approach consists of training one multi-class, and possibly also multi-label, classifier for each level of the label hierarchy. When a new test sample is presented the output of the classifiers from each level is used as the final classification.

This approach, however, is prone to label inconsistency, as different classifiers are trained for each level of the hierarchy and should, therefore, also be complemented by a post-processing method to correct the prediction inconsistency.

One common problem for all local classifier strategies the utilize the top-down class-prediction approach is the propagation of errors down the hierarchy.

3.2 Global Classifier

Another type of strategy is to learn a classifier than can globally learn to output the predictions for each level in the hierarchical structure. This is done by flattening the whole hierarchical structure.

Having only a single classifier, although easier to tune, it can turn the hierarchical classification into a much harder problem, specially having a sparse label space with an order of magnitude of 10^2 , i.e. there are 343 possible classes, but the labels co-occurrence can be a good guiding heuristic for a statistical model to infer the hierarchical label structure associated with a given sample.

4 Systems Developed

We developed two systems implementing the following approaches:

Local Classifier: a classifier per parent node using different types of classifiers;

Global Classifier: a single classifier relying on the hypothesis to explore the labels co-occurrence;

4.1 Local Classifier

We employed a classifier per parent node approach, which has the advantage of not being prone to label inconsistency errors. We need to train classifiers for each parent node. For Level 3 we don't need to train any classifier since it contains only leaf nodes, plus some nodes on Level 2 are already leaf nodes.

According to Table 1 the Level 1 has 8 possible labels, which means that the parent node of the first level (i.e, the Root Node) needs to be trained in a multi-label fashion and predict over 8 classes. Each of these 8 classes represents a parent node of some child classes on the next level of the hierarchy. So for Level 1 we need to train eight multi-label classifiers where the labels are the child's of each parent in the root level. Finally, for Level 2, we train only 42 classifiers, since according to the hierarchy-tree some labels in this level are already leaf nodes, and only 42 labels have then child labels. So, in total we trained 51 classifiers distributed by different levels as described in Table 4.

Level	Nr. Parent Nodes
Root Node	1
Level 1	8
Level 2	42
Total Classifiers	51

Table 4: Number of parent nodes per level in the hierarchy.

As stated before, one of the advantages of this approach is that it always produces a label structure that is enforced by the hierarchy-tree, but it is prone to error propagation from the top levels further down the tree.

4.2 Global Classifier

The global classifier needs to be a multi-label classifier targeting a label space with a total of 343 classes. One of the advantages is that there is only one single multi-label classifier to tune and explore, but on the other hand it has a high and sparse label space. Plus, one needs to employ

some post-processing cleaning to enforce the hierarchical structure, since there is an hierarchical dependency between the some of the 343 possible classes. For instance, the classifier can predict the labels 4.3 and 4.4 - corresponding to labels in the Level 2 in the hierarchy - but not the label 4, corresponding to Level 1 in the hierarchy.

5 Experiments and Results

In this section we describe the experimental setup and results for the two devised strategies for tackling both sub-tasks.

5.1 Results on the Development Set

During the development phase we only had the labels for training, so we randomly split the training dataset into two-subsets of 70% and 30% for training and parameter tuning, respectively. Then, train on the whole dataset with the best parameters and using the model to generate predictions on the development dataset. This approach was mainly to have a working framework for experiments and submit valid results.

During the test phase the labels for the development set were made available and we could use them to tune the classifiers. We used 3-fold cross-validation to perform parameter search using the training dataset. The parameter configuration which yielded the best results was then used to train the classifiers over the complete training dataset, and the classifier is then evaluated against the development set. Results reported in this section are all in regard to the development set.

5.1.1 Pre-processing

For representing a book we concatenated the book's title with book's textual description. In some cases only the title is present, for this cases we simply use the title.

We explored two tokenization schemas, one tokenizes the blurbs into sentences, and then from sentences into tokens, considering the title of the book as a sentence, this tokenization strategy was based on the german sentence tokenizer, and the *word_punkt_tokenizer* from NLTK 3.4.1 (Bird et al., 2009), and considered alphanumeric tokens only. The other approach was based on a simple regular expression: `(?u) \b\w\w+\b`. We also experimented lower casing and removing stop-words. After running a few experiments and comparing some initial results we opted for the regular

expression for tokenization, lower case token representations and removal of stop-words. For the neural networks the padding was done to match the size of the longest document in the dataset.

5.1.2 Prediction threshold gltment

We set the prediction threshold to 0.5, so any label with a predicted probability above 0.5 is selected. We noticed that for some samples no labels were being selected, as all labels had predicted probability scores lower than 0.5. To tackle this problem, for a given sample, if no predictions were done we lowered the threshold to 0.4, if still no label predictions are done, we lowered again the threshold to 0.3. This was done in a simple ad-hoc way, and no proper strategy was employed, and is done in a per label fashion.

5.1.3 Models implementation

For all the neural network models we used pre-trained embeddings, specifically the public available German fastText embeddings trained on Wikipedia, of dimension 300 and obtained using the skip-gram model as described in Bojanowski et al. (2017), the embeddings are fine-tuned during learning and out of vocabulary words are randomly initialized.

The training was done with the Adam optimizer (Kingma and Ba, 2014) using binary cross-entropy as a loss function and 30% of the training dataset for validation. Unless stated otherwise training was performed with mini-batch sizes of 16 for 10 epochs. All the neural network models were implemented in Keras 2.2.4 with Tensorflow 1.13.1 backend. The Logistic Regression classifier was based on the scikit-learn 0.21.1 (Pedregosa et al., 2011).

All the code used for this experiments is available on-line ¹.

5.1.4 Local classifier per node: Root Node

We explored different classifiers for the Root Node and for Level 1 and Level 2 we selected a Convolutional Neural Network (CNN).

We briefly describe the models used for the Root Node and the parameters explored, in bold we have the parameters that yielded the best scores:

Logit (TF-IDF): a logistic regression classifier with TF-IDF weighted vectors in a one-

¹https://github.com/davidsbatista/GermEval-2019-Task_1

versus-rest scenario varying the following parameters:

- n-grams: 1, **2**, 3;
- class weight: **balanced**, not-balanced;
- norm: 11, **12**;
- regularization C : 0.1, 10, 100, **300**;

Training was performed with Stochastic Average Gradient for a maximum of 5 000 iterations.

CNN (Kim, 2014): for sentence classification with rectified linear units in the activation functions of the 1D convolutions, and with a fully connected layer of size 600 and varying:

- filter windows: (1), **(1, 2)**, (1, 2, 3);
- feature maps: 256, **300**;

LSTM (Hochreiter and Schmidhuber, 1997): recursively reads each token in the text updating it's internal state and using the last state to represent the document, with a dropout layer of rate 0.1 between the LSTM's last state and the final sigmoid layer.

- single LSTM vs **bi-directional LSTM**;
- hidden units: 32, 64, **128**;

Bag-of-Tricks (Joulin et al., 2017): token embeddings representations are averaged into a single vector representation, which is fed to a sigmoid classifier, we varied the following parameters:

- n-grams: 2, 3, 4, 5;
- top-k most frequent tokens: **100k**, 90k, 80k;

Table 5 shows the results for different classifiers when trained with the best parameters on the training set and evaluated against the development set.

Method	Precision	Recall	F ₁
Logit (TF-IDF)	0.8211	0.8359	0.8284
CNN	0.8542	0.7879	0.8197
bi-LSTM	0.8062	0.7987	0.8024
Bag-of-Tricks	0.3787	0.6717	0.4843

Table 5: Results for different classifiers on the Sub-Task A on the development set.

The Logistic Regression classifier achieved the best results although the CNN classifier had a similar F₁ score, essentially by trading recall for precision.

5.1.5 Local classifier per node: Level 1 and 2

For Level 1 and 2 in the hierarchy tree we trained a total of 50 classifiers for each parent node, 8 for Level 1 and 42 for Level 2. All these classifiers were based on the CNN model.

We explored some parameters configurations by varying the filter windows size and the filter maps size, Table 6 shows a subset of the different configuration parameters tried which yielded some of the best results. All these classifiers were training with mini-batch size of 16 for 5 epochs.

	Filter Windows	Filter Maps
Conf ₁	1,2	300
Conf ₂	1,2,3	200
Conf ₃	1,2,3,5,7	300
Conf ₄	3,5,6,10	256

Table 6: Different configuration parameters for the CNN classifiers for Level 1 and 2.

In these experiments we used the Root Node classifier which yielded the best results, i.e. the logistic regression, to predict the labels for the root level, and experiment with different parameters for the Levels 1 and 2. Table 7 shows the results for Sub-Task B for configurations of parameters in Table 6.

	Precision	Recall	F ₁
Conf ₁	0.7151	0.5330	0.6108
Conf ₂	0.7144	0.5303	0.6087
Conf ₃	0.7219	0.5235	0.6069
Conf ₄	0.7274	0.5085	0.5986

Table 7: Results for Sub-Task B for different configurations of the CNN-based classifier for Levels 1 and 2 of the hierarchy, using the best classifier for the Root Level from Table 6.

We opted not to use a logistic regression classifier for Level 1 and 2 since this type of classifier needed to be trained in a one-versus-rest fashion. This would mean that for Level 2 we would need to train 93 classifiers and 242 for Level 3.

5.1.6 Global classifier

The global classifier uses a flattened hierarchy and learns how to predict a vector of 343 dimensions.

One advantage of this approach, in contrast to the local one, is the we need only to tune a single classifier, and both sub-tasks can also be tackled with this single classifier.

Our initial idea was to use a neural network architecture and leverage on the labels co-occurrence by initializing a weight matrix - we describe this idea in Section 6 - but due to time constraints this was not possible to explore this idea to the end. Also, due to time constraints we did not employ a post-processing step.

Nevertheless, we still applied a CNN architecture and explored different configuration parameters, varying the filter windows and the filter maps.

We used the same tokenization schema as with the Local classifier, as well as the same pre-trained word embeddings, which are fine-tuned during training.

The training was done with the Adam optimizer (Kingma and Ba, 2014) using binary cross-entropy as a loss function, with mini-batch sizes of 128 for 250 epochs and using 30% of the training dataset for validation.

We also used the same threshold filtering strategy as described in Section 5.1.2.

Table 8 presents some of the configurations of parameters used in the experiments and Table 9 the corresponding results for the same configurations for both sub-tasks.

	Filter Windows	Filter Maps
Conf ₁	1, 2, 3	300
Conf ₂	3, 5, 7, 10	256
Conf₃	1, 2, 3, 5, 7, 10	256

Table 8: Different configuration parameters for the CNN classifiers for the global classifier.

	Precision	Recall	F₁
Conf₁			
Sub-Task A	0.7163	0.7484	0.7320
Sub-Task B	0.5257	0.4603	0.4909
Conf₂			
Sub-Task A	0.7353	0.7686	0.7516
Sub-Task B	0.5470	0.4717	0.5066
Conf₃			
Sub-Task A	0.8389	0.7659	0.8008
Sub-Task B	0.6733	0.5032	0.5760

Table 9: Results for both sub-tasks using the configuration parameters from Table 8.

5.2 Test Results

We applied the classifiers described in the Section 5 with the parameters that yielded the best results on the development dataset, by training on all available data (i.e., training + development sets) and applied them on the test dataset, therefore generating two submissions for each sub-task.

5.2.1 Parent Per Node

With the one parent per node strategy classifier, we achieved the 13th best place on Sub-Task A, and the 11th best place on Sub-Task B, out of a total of 19 submissions. Results for the detailed evaluation metrics are described in Table 10.

This classifier achieved the 9th best recall and the 15th best precision for Sub-Task A, and the 8th best recall and the 14th best precision for Sub-Task B.

For roughly 5% of the test samples the classifier did not produce any predictions, due to the class probabilities scores being lower 0.3., the lowest possibly threshold selected in the adjustment.

Task	Precision	Recall	F ₁
Sub-Task A	0.8144	0.8255	0.8199
Sub-Task B	0.7042	0.5274	0.6031

Table 10: Best achieved results on the development set for both Sub-Tasks A and B with the parent per node classifier.

5.2.2 Global

With the global classifier strategy we achieved the 17th best place on Sub-Task A, and the 13th best place on Sub-Task B. Detailed results are presented in Table 11.

Task	Precision	Recall	F ₁
Sub-Task A	0.7761	0.7839	0.7839
Sub-Task B	0.5672	0.5185	0.5418

Table 11: Best achieved results on the development set for both Sub-Tasks A and B with the global classifier.

This classifier achieved the 15th best recall and the 17th best precision for Sub-Task A, and the 9th best recall and the 18th best precision for Sub-Task B. For roughly 1% of the samples the classifier did not produce any predictions. The hierarchy consistency is of 0.9363, reflecting the lack of a post-processing step to enforce the hierarchical structure, which would be 1.0 if employed.

Considering only the best submissions from all teams, we ranked 8th for Sub-Task A and 6th for Sub-Task B.

6 Future Work

We had planned to explore different features and carry more experiments but time constraints for the submission of the test results did not allow us to experiment all that was planned.

One crucial aspect in the global classifier is a post-processing step to make sure that the labels output is in-line with the hierarchy-tree constrains. The global classifier could also be improved by initializing a weight matrix based on label co-occurrence. Kurata et al. (2016) proposed a neural network initialization method to treat some of the neurons in the final hidden layer as dedicated neurons for each pattern of label co-occurrence. These dedicated neurons are initialized to connect to the corresponding co-occurring labels with stronger weights than to others representing non co-occurring labels. Baker and Korhonen (2017) applied this idea in the biomedical domain and to a much more compact hierarchy than the one presented in this paper.

In the local classifier strategy for Level 1 and 2 we use the same architecture for all classifiers and tuned them in the same way, the type of architecture and tuning process could be made dependent on the numbers of samples available to train and level in the hierarchical tree.

A few more features could have been explored, for instance the author’s name and the release date of the book. The padding of the documents representation for the neural network could be set to the average since of the documents, instead of the longest one.

The values for the prediction threshold were selected in an ad-hoc fashion, these could also be properly set, through a set of experiments.

References

- Simon Baker and Anna Korhonen. 2017. [Initialzing neural networks for hierarchical multi-label text classification](#). In *BioNLP 2017*, pages 307–315, Vancouver, Canada,. Association for Computational Linguistics.
- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*, 1st edition. O’Reilly Media, Inc.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. [Enriching word vectors with subword information](#). *Transactions of the Association for Computational Linguistics*, 5:135–146.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. [Long short-term memory](#). *Neural Comput.*, 9(8):1735–1780.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. [Bag of tricks for efficient text classification](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Yoon Kim. 2014. [Convolutional neural networks for sentence classification](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. [Adam: A method for stochastic optimization](#). *CoRR*, abs/1412.6980.

Daphne Koller and Mehran Sahami. 1997. [Hierarchically classifying documents using very few words](#). In *Proceedings of the Fourteenth International Conference on Machine Learning, ICML ’97*, pages 170–178, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes. 2017. [Hdltex: Hierarchical deep learning for text classification](#). In *ICMLA*, pages 364–371. IEEE.

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. [Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 521–526, San Diego, California. Association for Computational Linguistics.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. [Rcv1: A new benchmark collection for text categorization research](#). *J. Mach. Learn. Res.*, 5:361–397.

Carolyn E Lipscomb. 2000. Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88(3):265.

Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artières, George Palouras, Éric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Gallinari. 2015. [LSHTC: A benchmark for large-scale text classification](#). *CoRR*, abs/1503.08581.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. **Scikit-learn: Machine learning in python.** *J. Mach. Learn. Res.*, 12:2825–2830.

Steffen Remus, Rami Aly, and Chris Biemann. 2019. Germeval-2019 task 1: Shared task on hierarchical classification of blurbs. In *Proceedings of the GermEval 2019 Workshop*, KOVENS ’19, Erlangen, Germany.

Carlos N. Silla, Jr. and Alex A. Freitas. 2011. **A survey of hierarchical classification across different application domains.** *Data Min. Knowl. Discov.*, 22(1-2):31–72.

Jônatas Wehrmann, Rodrigo C. Barros, Silvia N. das Dôres, and Ricardo Cerri. 2017. **Hierarchical multi-label classification with chained neural networks.** In *Proceedings of the Symposium on Applied Computing*, SAC ’17, pages 790–795, New York, NY, USA. ACM.

Convolutional Neural Networks for Classification of German Blurbs

Erdan Genc, Louay Abdelgawad, Viorel Morari, Peter Kluegl

Averbis GmbH, Freiburg, Germany

{firstname}.{lastname}@averbis.com

Abstract

This paper presents the submission of the system AVERBIS_BOHB_CNN for the Shared Task on hierarchical classification of German blurbs (short texts) - GermEval 2019 Task 1. We optimized the hyperparameters of a CNN based on a fastText word embedding layer and combined it with a variant of a T-Criterion classification method. The model was able to achieve an F1 score of 0.834 (ranked 6th) for subtask a and of 0.644 (ranked 3th) for subtask b on the respective test sets.

1 Introduction

Hierarchical multi-label text-classification (HMC) is the task of classifying text into categories with an underlying hierarchical structure. As more and more text becomes available in digital form the need for such automated and robust text classification grows bigger. To foster research in the domain of HMC the organizers of GermEval 2019 called for participation in Shared Task 1 - hierarchical classification of German blurbs. In this work, we describe our submission to the task.

The task consists of two subtasks (subtask_a, subtask_b) in which the participants are challenged to classify short German text snippets advertising books into one or multiple of 8 non-hierarchically (a) and 343 (b) hierarchically structured categories, respectively.

We approach the subtasks with a convolutional neural net (CNN) (Kim, 2014), using word embeddings trained with fastText (Bojanowski et al., 2017). In order to optimize the involved hyperparameters, we used BOHB (Falkner et al., 2018). BOHB is a hyperparameter search technique, which combines Bayesian optimization with bandit-based methods to find good configurations in feasible time. The hierarchy of labels in

subtask_b was not incorporated in the learning process, i.e. the labels' hierarchical structure was flattened. To further improve the predictions based on the probability distributions, we adapted the T-Criterion (Boutell et al., 2004) classification method. The approach reaches F1 scores of 0.850 on the validation set of subtask_a and 0.664 on subtask_b.

The rest of the paper is structured as follows. The next Section (2) describes the provided dataset. Section 3 introduces the preliminaries. In Section 4, the system description is layed out. Section 5 illustrates the results and Section 6 summarizes the learnings and gives an outlook for future work.

2 Dataset

The dataset is a collection of short German text sequences, so called blurbs, advertising German books. Figure 1 shows an example of a blurb. Each instance features different fields such as *title*, *body*, *copyright*, *categories*, *authors*, *published*, *isbn*, *url*. The *body* is the main text field and contains the advertising description.

The *categories* are the target classes of the classification tasks. Each blurb can be classified into one or multiple *categories*. Each *category* consists of one or many hierarchically structured *topics*.

For subtask_a the blurbs need to be classified into one or multiple of the eight first level classes, i.e. root level topics ($d = 0$):

- Literatur & Unterhaltung
- Sachbuch
- Kinderbuch & Jugendbuch
- Ratgeber
- Ganzheitliches Bewusstsein
- Glaube & Ethik
- Künste
- Architektur & Garten

For `subtask_b`, the blurbs need to be classified into multiple hierarchical classes, i.e. topics ($d \in \{1, 2\}$). In total there are 343 topics with a maximum level depth of three.

The complete dataset contains 20,784 instances of German blurbs. 14,548 labeled instances for training (`train`) as well as 2,079 labeled instances for local validation (`dev`). All results in this work are based on these two sets. A third set of 4,157 unlabeled instances was made available for result submission (`test`).

```
<book date="2019-01-04" xml:lang="de">
<title>Das letzte Kind</title>
<body>Es ist ein Jahr ... spannende Geschichte.</body>
<copyright>(c) Verlagsgruppe Random House GmbH</copyright>
<categories>
<category>
<topic d="0">Literatur & Unterhaltung</topic>
<topic d="1">Krimi & Thriller</topic>
<topic d="2" label="True">Psychothriller</topic>
</category>
</categories>
<authors>John Hart</authors>
<published>2010-08-13</published>
<isbn>9783641048198</isbn>
<url>https://www.randomhouse.de/ebook/.../.rhd%0A</url>
</book>
```

Figure 1: Example blurb taken from the `dev` set.

With concatenated fields, the average length of a blurb is 231 words with a standard deviation of 64. The smallest blurb counts 66 words, the largest 1017.

The distribution of classes in the `train` set for `subtask_a` are shown in Figure 2. As we can see, the distribution is highly imbalanced with more than half of the instances being labeled with the topic *Literatur & Unterhaltung*.

3 Preliminaries

This section introduces the preliminaries. The support vector machine that serves as baseline in Subsection 3.1, word embeddings (3.2), convolutional neural networks (3.3) and hyperparameter optimization (3.4).

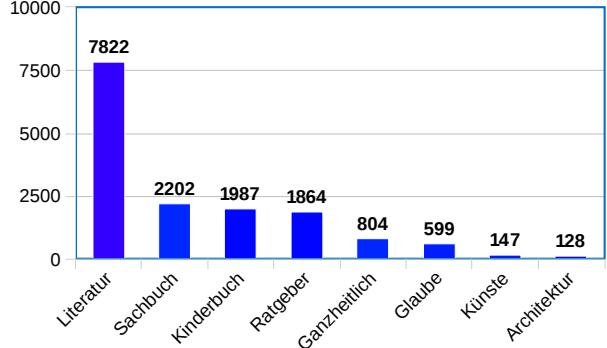


Figure 2: Number of samples per class for `subtask_a` in the `train` set.

3.1 SVM

Support Vector Machines (SVM) are a well established machine learning approach for text classification (Manevitz and Yousef, 2001). In this work, the multi-label classifier is implemented by several one-vs-all linear SVMs (Fan et al., 2008). The features of the single SVMs consist of a bag-of-stems¹. The stems are prefixed to encode the information to which field it belongs to, e.g. `title_stem` or `body_stem`. The features are weighted using logarithmic frequencies with redundancy (Leopold and Kindermann, 2002) and are L2-normalized. At prediction time the T-Criterion classification method with a threshold of 0.5 is used for all labels. Opposed to state-of-the-art neural classifiers in which the input is encoded using distributed representations for words, e.g. word embeddings, SVMs with a bag-of-words feature representation neglect all sequential information.

3.2 Word Embeddings

A word embedding describes a mapping that translates single words or phrases taken from a vocabulary into an n -dimensional real-valued vector space. Using the context of the words, it is usually the rationale to find a representation which preserves syntactic and semantic attributes and can be passed on to a machine learning algorithm.

The choice of an effective word embedding depends on a large variety of parameters, e.g. the model itself, embedding size, the corpora used for training or the action taken for unknown words. Many standardized and well-described word embeddings, trained on different corpora, can be found online.

¹<https://snowballstem.org/>, accessed September 18, 2019

For this work, fastText (Bojanowski et al., 2017) was selected as it additionally considers n-gram sub-words as input instead of whole words as atomic units. Apart from the increased training time, it enables the model to embed unseen words. The word "Propene", for example, might be unseen in the training corpus, yet fastText is able to assign a vector near the vector of the seen word "Propylene" which can be advantageous in domains with complex language.

3.3 CNN

Early improvements in text classification with deep learning started with the Dynamic Convolutional Neural Networks (DCNN) method (Kalchbrenner et al., 2014). The authors first adopted CNNs, a model well-received in the computer vision domain, to the field of NLP. Following this work, Yoon Kim created another CNN architecture (Kim, 2014). The main improvement was to embed the input words using pre-trained word embeddings (Mikolov et al., 2013) before passing them into the neural network. Contrary to other convolutional networks, Kim's CNN uses a single stage of wide parallel convolutions instead of several stacked convolutions on top of each other. This architecture has been selected as main approach in this work as it has been proven to be versatile and efficient.

3.4 Hyperparameter Optimization

In general the parameters of a neural network can be divided into two categories: the normal and the hyperparameters. Normal parameters, such as weights and biases, are changed during training time. Hyperparameters, such as learning rate and batch size, are set before the training begins. The right choice of parameters can effect the performance of a neural network significantly (Henderson et al., 2017). Therefore, hyperparameter optimization aims to determine a set of hyperparameter values such that a objective function is maximized. These techniques range from simple random search to more sophisticated, efficient methods such as Bayesian Optimization (BO). In this work, BOHB, a state-of-the-art hyperparameter optimization technique developed by Falkner et al., is used to tune the parameters. It combines the best of two worlds leveraging the strong performance of BO while maintaining the speed of hyperband (HB).

4 System Description

This section presents a detailed description of the used system and how the experiments were conducted. It introduces the preprocessing steps, used word embeddings, the model itself as well as the hyperparameter optimization steps and the used classification methods that turn the probability distributions into predictions.

4.1 Preprocessing

This subsection describes the applied preprocessing steps for the experiments. The different fields of the blurbs are first concatenated and then tokenized using JTOK². The text is normalized by lower-casing, removing all non-alphabetic characters and reducing all multi-spaces to a single white space. As the types of CNNs used in this work require a constant sized input the blurbs are truncated/padded to a fixed length. This sequence length is optimized as a hyperparameter.

4.2 Word embeddings

As mentioned before, fastText (Grave et al., 2018) was used to obtain the word embeddings. In this work the pre-trained German model is used.³ The number of words, i.e. the top-n most frequent words in the dataset that are embedded, is optimized as a hyperparameter.

4.3 Model

The main focus of this work relies on optimizing CNNs for the task of classifying German text blurbs. The preprocessed text is fed to the network through the earlier described word embedding lookup, which converts word IDs to vectors represented in a high-dimensional vector space.

Afterwards, a convolutional layer is applied on top of the embedding layer. The layer is specified by two central hyperparameters: region sizes and number of filters.

A region size can be understood as the 1-D convolution window size in the domain of computer vison or the n-gram size in the domain of NLP. Each region has a number of filters with different weights. The weights of each filter are adjusted during training time to detect different lexical features in the text.

²<https://github.com/DFKI-MLT/JTok>, accessed July 30, 2019

³<https://fasttext.cc/docs/en/crawl-vectors.html>, accessed August 05, 2019

In order to reduce the output of the different filters at each position in the text, max-over-time pooling is applied. The pooling returns the highest value for each filter with respect to all positions; i.e. after the pooling step, a layer with $|region\ sizes| \cdot number\ of\ filters$ neurons is obtained. These neurons are fully connected to the final sigmoid layer which turns the input into a probability distribution over the label classes.

In order to mitigate overfitting, a Dropout rate is established between these last two layers. This way the network is forced to not solely rely on the activation of single filters.

Besides the CNN’s specific hyperparameters, it is also recommended to optimize the learning rate and batch size.

To adjust the weights during learning, Adam optimization (Kingma and Ba, 2015) is used.

4.4 Applied Hyperparameter Optimization

In this section, the results of applying Hyperparameter Optimization by using BOHB (Falkner et al., 2018) are investigated. The hyperparameters are optimized by sampling a random 20% train/test split at the beginning of each run, which is necessary to mitigate overfitting. In total BOHB evaluated 30 iterations to find the final configuration. The search hyperparameter space is shown in Table 1. The final hyperparameters are shown in Table 2. The parameters are very different for both subtasks. It can be seen that the best parameters for the fine-grained classification of subtask_b span a more complex network than for subtask_a which intuitively makes sense. The sequence length is more than twice as long ($191 \rightarrow 410$), the number of words almost reaches the upper limit of the range ($29,524 \rightarrow 48,736$) and the number of filters per region is also highly increased ($560 \rightarrow 975$). The region size on the other hand drops ($15 \rightarrow 7$), which may indicate that smaller text sequences play a more important role for the fine-grained classification.

4.5 Classification Methods

A special classification method based on the T-Criterion (Boutell et al., 2004) was used to further improve the classification results based on the probability distribution over the label classes. In the standard above threshold classification method, all class probabilities with a value above a pre-set threshold (0.5), are considered as posi-

Table 1: Hyperparameter search ranges.

Parameter	Range
<i>Sequence Length</i>	[50; 1,000]
<i>Number of Words</i>	[10,000; 50,000]
<i>Regions Size</i>	[3; 18]
<i>Number of Filters</i>	[200; 2,000]
<i>Dropout Rate</i>	[0.0; 0.7]
<i>Learning Rate</i>	[$1e^{-4}$; $5e^{-2}$]
<i>Batch Size</i>	[16; 256]

Table 2: Best found hyperparameter configurations for both subtasks.

Parameter	subtask	
	a	b
<i>Sequence Length</i>	191	410
<i>Number of Words</i>	29,524	48,736
<i>Regions Size</i>	[15]	[7]
<i>Number of Filters</i>	560	975
<i>Dropout Rate</i>	0.10	0.03
<i>Learning Rate</i>	0.0017	0.0022
<i>Batch Size</i>	32	64

tive classes. There are two problems with this approach.

First, given the Closed World Assumption that every blurb belongs to at least one class, if there is not a single label with a confidence greater than 0.5, there will be no prediction. This problem is solved by using the T-Criterion, if all confidences are lower than the pre-set threshold the label with the highest confidence is assigned if the overall confidence entropy is above a minimal threshold (0.1).

Second, as we flatten the hierarchy structure of subtask_b for the training of the CNN, using T-Criterion alone may result in positive classes for a single node in the hierarchy. Therefore, a reconstruction step is applied which additionally predicts the classes in the path from root to predicted node. This classification method will be described as T-Criterion.

5 Results

Table 3 illustrates the results for both subtasks per approach based on the dev and test set. As expected, the F1 scores for subtask_a (0.850) are higher than the scores for the HMC subtask_b (0.664) with 343 classes. The

Approach	dev		test	
	subtask_a	subtask_b	subtask_a	subtask_b
SVM	0.783	0.577	-	-
AVERBIS_BOHB_CNN	0.837	0.641	-	-
AVERBIS_BOHB_CNN + T-Criterion	0.850	0.664	0.834	0.644

Table 3: F1 scores for both subtasks per approach.

AVERBIS_BOHB_CNN outperforms the baseline SVM approach (+0.054). Moreover, applying the T-Criterion classification method further improves AVERBIS_BOHB_CNN (+0.013). Unfortunately, the scores deteriorate comparing dev to test set. This could indicate overfitting on the test + dev set and therefore question the reliability of the found hyperparameters. For more robust results we suggest to tweak the hyperparameter ranges and increase the iteration budget of the optimization.

6 Conclusion

In conclusion we have shown that a strong classifier for German blurbs can be build using a CNN with optimized hyperparameters. Yet, especially the HMC subtask.b is a challenging problem that requires further work. The found hyperparameters for both subtasks illustrate how the complexity of a CNN, i.e. number of trainable parameters, grows from a simple text-classification task with 8 labels to the hierarchical task with 343 labels, given the same input.

As deployment into production is an important factor for us, we plan to compare our results to more recent approaches in terms of accuracy, training and inference time; e.g. Acharya et al. report on decreased model sizes and inference times while maintaining high accuracy by compressing the word embeddings.

References

- Anish Acharya, Rahul Goel, Angeliki Metallinou, and Inderjit Dhillon. 2019. [Online embedding compression for text classification using low rank matrix factorization](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:6196–6203.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. 2004. [Learning multi-label scene classification](#). *Pattern Recognition*, 37(9):1757 – 1771.
- Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. [BOHB: robust and efficient hyperparameter optimization at scale](#). *CoRR*, abs/1807.01774.
- Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.
- Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.
- Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2017. [Deep reinforcement learning that matters](#). *CoRR*, abs/1709.06560.
- Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Edda Leopold and Jörg Kindermann. 2002. Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, 46(1-3):423–444.
- Larry M Manevitz and Malik Yousef. 2001. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

TwistBytes - Hierarchical Classification at GermEval 2019: walking the fine line (of recall and precision)

Fernando Benites

benf@zhaw.ch

Zurich University of Applied Sciences,
Switzerland

Abstract

We present here our approach to the GermEval 2019 Task 1 - Shared Task on hierarchical classification of German blurbs. We achieved the first place in the hierarchical subtask B and second place on the root node, flat classification subtask A. In subtask A, we applied a simple multi-feature TF-IDF extraction method using different n-gram range and stop-word removal, on each feature extraction module. The classifier on top was a standard linear SVM. For the hierarchical classification, we used a local approach, which was more light-weighted but was similar to the one used in subtask A. The key point of our approach was the application of a post-processing to cope with the multi-label aspect of the task, increasing the recall but not surpassing the precision measure score.

1 Introduction

Hierarchical Multi-label Classification (HMC) is an important task in Natural Language Processing (NLP). Several NLP problems can be formulated in this way, such as patent, news articles, books and movie genres classification (as well as many other classification tasks like diseases, gene function prediction). Also, many tasks can be formulated as hierarchical problems in order to cope with a large amount of labels to assign to the sample, in a divide and conquer manner (with pseudo meta-labels). A theoretical survey exists Silla and Freitas (2011) discussing how the task can be engaged, several approaches and the prediction quality measures. Basically, the task in HMC is to assign a sample to one or many nodes of a Directed Acyclic Graph (DAG) (in special cases a tree) based on features extracted from the sample. In the case of possible multiple parent, the evaluation of the prediction complicates heavily, for once since several paths can be taken, but only in a joining node must be considered.

The GermEval 2019 Task 1 - Shared Task on hierarchical classification of German blurbs focus on the concrete challenge of classifying short descriptive texts of books into the root nodes (subtask A) or into the entire hierarchy (subtask B). The hierarchy can be described as a tree and consisted of 343 nodes, in which there are 8 root nodes. With about 21k samples it was not clear if deep learning methods or traditional NLP methods would perform better. Especially, in the subtask A, since for subtask B some classes had only a few examples. Although an ensemble of traditional and deep learning methods could profit in this area, it is difficult to design good heterogeneous ensembles.

Our approach was a traditional NLP one, since we employed them successfully in several projects Benites (2017); Benites and Cieliebak (2017); Benites et al. (2019), with even more samples and larger hierarchies. We also compared new libraries and our own implementation, but focused on the post-processing of the multi-labels, since this aspect seemed to be the most promising improvement to our matured toolkit for this task. Therefore, we aimed to push recall up and hoped to not overshoot much over precision.

2 Related Work

The dataset released by Lewis et al. (2004) enabled a major boost in HMC on text. This was a seminating dataset since it not only was very large (800k documents) but the hierarchies were large (103 and 364). Many different versions were used in thousands of papers. Further, the label density Tsoumakas and Katakis (2007) was considerably high allowing also to be treated as multi-label, but not too high as to be disregarded as a common real-world task. Some other datasets were also proposed (Partalas et al. (2015), Mencía and

Fürnkranz (2010)), which were far more difficult to classify. This means consequently that a larger mature and varied collection of methods was developed, from which we cannot cover much in this paper.

An overview of hierarchical classification was given in Silla and Freitas (2011) covering many aspects of the challenge. Especially, there are local approaches which focus on only part of the hierarchy when classifying. They are in contrast to the global (big bang) approaches.

A difficult open problem relates to the selection of which hierarchical quality prediction measure to use since there are dozens of them. An overview with a specific problem is given in Brucker et al. (2011). An approach which was usually taken was to select several measures, and use a vote, although many measures inspect the same aspect and therefore correlate, creating a bias. The GermEval competition did not take that into account and concentrates only on the flat micro F-1 measures¹.

Still, a less considered problem in HMC is the number of predicted labels, especially regarding the post-processing of the predictions². We discussed this thoroughly in Benites (2017). The main two promising approaches were proposed by Yang (1999) and Read et al. (2009). The former focuses on column and row based methods for estimating the appropriate threshold to convert a prediction confidence into a label prediction. Read et al. (2009) used the label cardinality (Tsoumakas and Katakis (2007)), which is the mean average label per sample, of the training set and change the threshold globally so that the test set achieved similar label cardinality.

3 Data and Methodology

3.1 Task Definition and Data Description

The shared task aimed at Hierarchical Multi-label Classification (HMC) of Blurbs. Blurbs are short texts consisting of some German sentences. Therefore, a standard framework of word vectorization could be applied. There were 14548 train-

¹The harmonic mean between micro recall and precision gives more weight for the predominant label. Many new tasks consider the macro averaged F-1 since it gives equal weights for all labels which can be interesting for a large amount of labels (or samples to come).

²This is especially important if macro F-1 is used as quality prediction measure, in order to predict as many labels as possible.

ing, 2079 development, and 4157 test samples.

The used hierarchy can be considered as an ontology, but for the sake of simplicity, we regard it as a simple tree, each child node having only one single parent node, with 4 levels of depth, 343 labels of which 8 are root nodes, namely: 'Literatur & Unterhaltung', 'Ratgeber', 'Kinderbuch & Jugendbuch', 'Sachbuch', 'Ganzheitliches Bewusstsein', 'Glaube & Ethik', and 'Künste, Architektur & Garten'.

The label cardinality (average number of labels per sample) of the training dataset was about 1.070 (train: 1.069, dev: 1.072) in the root nodes, pointing to a clearly low multi-label problem, although there were samples with up to 4 root nodes assigned. This means that the traditional machine learning systems would promote single label predictions. Subtask B has a label cardinality of 3.107 (train: 3.106, dev: 3.114), with 1 up to 14 labels assigned per sample. Table 1 shows a short dataset summary by task.

Task	samples	labels	cardinality	density
subtask A	20,784	8	1.069	0.1336
subtask B	20,784	343	3.11	0.0091

Table 1: Specs for dataset for subtasks A and B

3.2 System Definition

We used two different approaches for each subtask. In subtask A, we used a heavier feature extraction method and a linear Support-Vector-Machine (SVM) classifier. Whereas for subtask B, we used a more light-weighted feature extraction with the same SVM but in a local-hierarchical-classification fashion, i.e. for each parent node such a base classifier was used. Also the use of a different postprocessing step per task differentiate the approaches. They were designed to be light and fast, to work almost out of the box, and to easily generalise.

3.2.1 Classifiers

Base Classifier For subtask A, we use the one depicted in Fig. 1, for subtask B, a similar more light-weight approach was employed as base classifier (described later). As can be seen, several vectorizers based on different n-grams (word and character) with a maximum of 100k features and preprocessing, such as using stopwords or not, were applied to the blurbs. The obtained term frequencies were then weighted with inverse docu-

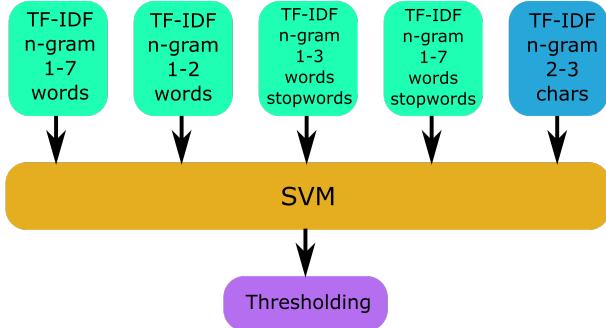


Figure 1: SVM-TF-IDF classifier with ensemble of textual features

ment frequency (TF-IDF). The results of five different feature extraction and weighting modules were given as input for a vanilla linear SVM classifier (scikit-learn LinearSVC) (parameter $C=1.5$) which was trained in an one-versus-all fashion.

3.2.2 Hierarchical Classifier

For the hierarchical task, we used a local parent node strategy, i.e. the parent node decided which of its children was assigned to the sample (one-vs-rest, in this case child versus siblings). This created also the necessity of a virtual root node. For each node the same base classifier is trained independently of the other nodes, so the amount of labels each classifier was confronted with was limited. We also adapted each feature extraction with the classifier in each single node much like Paes et al. (2014). As base classifier, a similar one to Fig. 1 was used, where only one 1-7 word n-gram, one 1-3 word n-gram with German stopwords removal and one char 2-3 n-gram feature extraction were employed, all with maximum 70k features, since it was performed for each parent node. We used two implementations achieving very similar results. In the following, we give a description of both approaches.

Recursive Grid Search Parent Node Our implementation is light-weighted and optimized for a short pipeline, nonetheless it is prepared for large amounts of data, saving each local parent node model to the disk. However, it does not conforms the way scikit-learn is designed. Further, in contrast to the Scikit Learn Hierarchical, we give the possibility to optimize with a grid search each feature extraction and classifier per node. This can be quite time consuming, but can also be heavily parallelized. In the final phase of the competition, we

did not employ it because of time constrains³ and the amount of experiments performed in the Experiments Section was only possible with a light-weighted implementation.

Scikit Learn Hierarchical Scikit Learn Hierarchical⁴ (Hsklearn) was forked and improved to deal better with multi-labels, which was a key feature of the shared task, as well as to allow each node to perform its own preprocessing⁵. This guaranteed that the performance of our own implementation was surpassed and that a contribution for the community was made. This ensured as well that the results are easily reproducible.

3.2.3 Post-processing: Threshold

Many classifiers can predict a score or confidence about the prediction. Turning this score into the prediction is usually performed by setting a threshold, such as 0 and 0.5, so labels which have a score assigned greater than that are assigned to the sample. This might not be the optimal threshold in the multi-label classification setup and there are many approaches to set it (Yang (1999)). Although these methods concentrate in the sample or label, we have had good results with a much more general approach.

As described in Benites (2017), Read and Pfahringer Read et al. (2009) introduce a method (referred hereinafter to as Label Cardinality Adjustment (LCA)) to estimate the threshold globally. Their method chooses the threshold that minimizes the difference between the label cardinality of the training set and the predicted set.

$$t = \underset{t \in [0,1]}{\operatorname{argmin}} |LCard(D_T) - LCard(H_t(D_S))|$$

where $LCard(D_T)$ denotes the label cardinality of training set and $LCard(H_t(D_S))$ the label cardinality of the predictions on test set if t was applied as the threshold. For that the predictions need to be normalized to unity⁶. We also tested this method not for the label cardinality over all

³The system was trained on a Intel Xeon 32 cores and 100 Gb RAM.

⁴<https://github.com/globality-corp/sklearn-hierarchical-classification/>

⁵<https://github.com/fbenites/sklearn-hierarchical-classification/>

⁶Although a sample wise normalization can be applied, we used a normalization over all predicted samples. This works especially good for Task A, since there is only one classifier at the top.

samples and labels but only labelwise. In our implementation, the scores of the SVM were not normalized, which produced slightly different results from a normalized approach.

For the HMC subtask B, we used a simple threshold based on the results obtained showing that on this task LCA performed worse (see Section 4.3). Especially, using multiple models per node could cause a different scaling and consequently making it difficult to use one threshold for all classifiers.

3.3 Alternative approaches

We also experimented with other different approaches. The results of the first two were left out (they did not perform better), for the sake of conciseness.

- Meta Crossvalidation Classifier: Benites et al. (2019)
- Semi-Supervised Learning: Jauhiainen et al. (2018); Benites et al. (2019)
- Flair: Flair Akbik et al. (2018) with different embeddings (BERT (out of memory)⁷, Flair embeddings (forward and backward German)). Such sophisticated language models require much more computational power and many examples per label. This was the case for the subtask A but subtask B was not.

4 Experiments

We divide this Section in three parts, in first we conduct experiments on the development set and in the second on the test set for Task A and in the third for Task B, in the latter two we also discuss the competition results.

4.1 Preliminary Experiments on Development Set

The experiments with alternative approaches, such as Flair, meta-classifier and semi-supervised learning⁸ yielded discouraging results, so we will concentrate in the SVM-TF-IDF methods. Especially, semi-supervised proved in other setups very valuable, here it worsened the prediction quality, so we could assume the same "distribution" of samples

⁷The system was trained on an Intel iCore 7 CPU with 32 Gb RAM with a NVIDIA GeForce 1060 6Gb GPU.

⁸The training, dev and test set seems to come from the same distribution, so the quality prediction when using a semi-supervised method was worse than without.

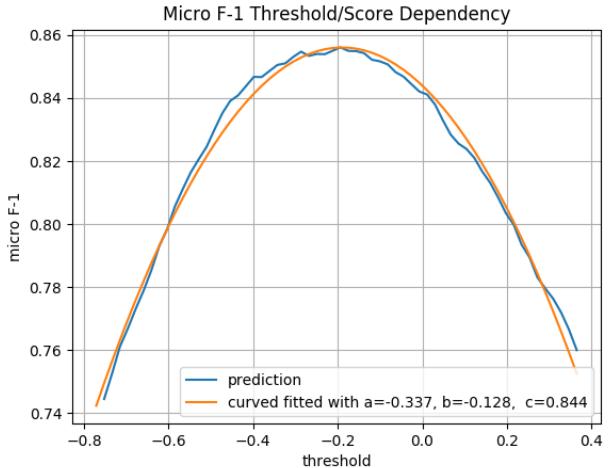


Figure 2: Threshold/micro F-1 dependency

were in the training and development set (and so we assume for the test set).

In Table 2, the results of various steps towards the final model can be seen. An SVM-TF-IDF model with word unigram already performed very well. Adding more n-grams did not improve the prediction quality, on the contrary using n-grams 1-7 decreased the performance. Only when removing stopwords it improved again, but then substantially. Nonetheless, a character 2-3 n-gram performed best between these simple models. This is interesting, since this points much more to not which words were used, but more on the morphology⁹.

Using the ensemble feature model produced the best results without post-processing. The simple use of a low threshold yielded also astonishingly good results. This indicates that the SVM's score production was already very good, yet the threshold 0 was too cautious.

In Fig. 2, a graph showing the dependency between the threshold set and the micro F-1 score achieved in the development set is depicted. The curve fitted was $a*x^2 + b*x + c$ which has the maximum at approx. -0.2. We chose -0.25 in the expectation that the test set would not have the exact characteristics as the development set and based on our previous experience with other multi-label datasets (such as the RCv1-v2) which produced best results at a threshold of -0.3. Also as we will see, the results proved us right achieving the best recall, yet not surpassing the precision score. This is a crucial aspect of the F-1 measure, as it is the

⁹For the sake of conciseness, we will not discuss it here.

Nr.	Method	micro F-1
1	SVM-TF-IDF, word unigram	0.7965
2	SVM-TF-IDF, word unigram, t=-0.25	0.8234
3	SVM-TF-IDF, word n-gram (1-7)	0.7875
4	SVM-TF-IDF, word n-gram (1-7), t=-0.25	0.8152
5	SVM-TF-IDF, word n-gram (1-3), stopwords	0.8075
6	SVM-TF-IDF, word n-gram (1-3), stopwords, t=-0.25	0.8240
7	SVM-TF-IDF, char n-gram (2-3)	0.8205
8	SVM-TF-IDF, char n-gram (2-3), t=-0.25	0.8332
9	SVM-TF-IDF, feat. ensemble	0.8414
10	SVM-TF-IDF, feat. ensemble, threshold LCA	0.8545
10	SVM-TF-IDF, feat. ensemble, threshold LCA normed	0.8534
11	SVM-TF-IDF, feat. ensemble, threshold LCA-labelwise	0.8603
12	SVM-TF-IDF, feat. ensemble, threshold -0.25	0.8540
13	SVM-TF-IDF, feat. ensemble, threshold -0.2	0.8557
14	Flair Embeddings German (forward,backward), 60 epochs	0.8151
15	SVM-TF-IDF, feat. ensemble, threshold LCA, fixing null	0.8577
16	SVM-TF-IDF, feat. ensemble, threshold LCA-labelwise, fixing null	0.8623

Table 2: Micro F-1 scores of different approaches on the development set classifying root nodes (subtask A), best four values marked in bold

harmonic mean it will push stronger and not linearly the result towards the lower end, so if decreasing the threshold, increases the recall linearly and decreases also the precision linearly, balancing both can consequently yield a better F-1 score.

Although in Fig. 2, the curve fitted is parabolic, in the interval between -0.2 and 0, the score is almost linear (and strongly monotone decreasing) giving a good indication that at least -0.2 should be a good threshold to produce a higher F-1 score without any loss.

Even with such a low threshold as -0.25, there were samples without any prediction. We did not assign any labels to them, as such post-process could be hurtful in the test set, although in the development it yielded the best result (fixing null).

In Table 3, the results are shown of the one-vs-all approach regarding the true negative, false positives, false negatives and true positives for the different threshold 0, -0.25 and LCA. Applying lower threshold than 0 caused the number of true positives to increase without much hurting the number of true negatives. In fact, the number of false positives and false negatives became much more similar for -0.25 and LCA than for 0. This results in the score of recall and precision being similar, in a way that the micro F-1 is increased without changing the scores of the prediction. Also, the threshold -0.25 resulted that the number of false

positive is greater than the number of false negatives, than for example -0.2. LCA produced similar results, but was more conservative having a lower false positive and higher true negative and false negative score.

We also noticed that the results produced by subtask A were better than that of subtask B for the root nodes, so that a possible crossover between the methods (flat and hierarchical) would be better, however we did not have the time to implement it. Although having a heavier feature extraction for the root nodes could also perform similar (and decreasing complexity for lower nodes). We use a more simple model for the subtask B so that the model would probably not overfit.

Table 4 shows the comparison of the different examined approaches in subtask B in the preliminary phase. Both implementations, Hsklearn and our own produced very similar results, so for the sake of reproducibility, we chose to continue with Hsklearn. We can see here, in contrary to the subtask A, that -0.25 achieved for one configuration better results, indicating that -0.2 could be overfitted on subtask A and a value diverging from that could also perform better. The extended approach means that an extra feature extraction module was added (having 3 instead of only 2) with n-gram 1-2 and stopwords removal. The LCA approach yielded here a worse score in the normalized but

Label	t=0				t=-0.25				LCA			
	tn	fp	fn	tp	tn	fp	fn	tp	tn	fp	fn	tp
Architektur & Garten	2062	0	4	13	2061	1	2	15	2061	1	2	15
Ganzheitliches Bewusstsein	1959	8	45	67	1951	16	29	83	1951	16	30	82
Glaube & Ethik	1986	3	31	59	1983	6	23	67	1984	5	24	66
Kinderbuch & Jugendbuch	1783	8	80	208	1759	32	50	238	1762	29	51	237
Künste	2061	0	6	12	2061	0	4	14	2061	0	4	14
Literatur & Unterhaltung	874	98	58	1049	801	171	31	1076	807	165	31	1076
Ratgeber	1799	20	110	150	1781	38	75	185	1785	34	77	183
Sachbuch	1701	40	148	190	1672	69	106	232	1674	67	111	227
Total	14225	177	482	1748	14069	333	320	1910	14085	317	330	1900

Table 3: Confusion matrix between label and others for threshold (t) =0 and =-0.25 (true negative: tp, false negative: fn, false positive: fp, true positive: tp)

Method	micro F-1
Hsklearn	0.6544
Hsklearn, t=-0.25	0.6758
Hsklearn, t=-0.2	0.6749
Hsklearn, LCA normalized	0.6645
Hsklearn, LCA	0.6717
Hsklearn extended	0.6589
Hsklearn extended, t=-0.25	0.6750
Hsklearn extended, t=-0.2	0.6765
own imp.	0.6541
own imp., t=-0.25	0.6704
own imp., t=-0.2	0.6715

Table 4: Preliminary experiments on subtask B, best three values marked in bold

almost comparable in the non-normalized. However, the simple threshold approach performed better and was therefore more promising.

4.2 Subtask A

In Table 5, the best results by team regarding micro F-1 are shown. Our approach reached second place. The difference between the first four places were mostly of 0.005 between each, showing that only a minimal change could lead to a place switching. Also depicted are not null improvements results, i.e. in a following post-processing, starting from the predictions, the highest score label is predicted for each sample, even though the score was too low. It is worth-noting that the all but our approaches had much higher precision compared to the achieved recall.

Despite the very high the scores, it will be difficult to achieve even higher scores with simple NLP scores. Especially, the n-gram TF-IDF with SVM could not resolve descriptions which are science fiction, but are written as non-fiction book¹⁰,

¹⁰Exemplary are books describing dystopias which from a

where context over multiple sentences and word groups are important for the prediction.

4.3 Subtask B

The best results by team of subtask B are depicted in Table 6. We achieved the highest micro F-1 score and the highest recall. Setting the threshold so low was still too high for this subtask, so precision was still much higher than recall, even in our approach. We used many parameters from subtask A, such as C parameter of SVM and threshold. However, the problem is much more complicated and a grid search over the nodes did not complete in time, so many parameters were not optimised. Moreover, although it is paramount to predict the parent nodes right, so that a false prediction path is not chosen, and so causing a domino effect, we did not use all parameters of the classifier of subtask A, despite the fact it could yield better results. It could as well have not generalized so good.

The threshold set to -0.25 shown also to produce better results with micro F-1, in contrast to the simple average between recall and precision. This can be seen also by checking the average value between recall and precision, by checking the sum, our approach produced $0.7072+0.6487 = 1.3559$ whereas the second team had $0.7377+0.6174 = 1.3551$, so the harmonic mean gave us a more comfortable winning marge.

5 Conclusion

We achieved first place in the most difficult setting of the shared Task, and second on the "easier" subtask. We achieved the highest recall and this score was still lower as our achieved precision (indicat-

n-gram perspective have very much the same vocabulary of a non-fiction book. Here, more aspects of the language need to be captured, such as a focus to constructions like "in a future New York City".

Rank	Teamname	precision	recall	micro F-1
1	EricssonResearch	0.8923	0.8432	0.8670
-	twistbytes LCA fixing null	0.8536	0.8790	0.8661
-	twistbytes LCA-labelwise fixing null	0.8536	0.8763	0.8648
2	twistbytes	0.8650	0.8617	0.8634
3	DFKI-SLT	0.8760	0.8472	0.8614
4	Raghavan	0.8777	0.8383	0.8575
5	knowcup	0.8525	0.8362	0.8443
6	fosil-hsmw	0.8427	0.832	0.8373
7	Averbis	0.8609	0.8083	0.8337
8	HSHL1	0.8244	0.8159	0.8201
9	Comtravo-DS	0.8144	0.8255	0.8199
10	HUIU	0.8063	0.8072	0.8067
11	LT-UHH	0.8601	0.7481	0.8002

Table 5: Results of subtask A, best micro F-1 score by team

Rank	Teamname	precision	recall	micro F-1
1	twistbytes	0.7072	0.6487	0.6767
2	EricssonResearch	0.7377	0.6174	0.6722
3	knowcup	0.7507	0.5808	0.6549
4	Averbis	0.677	0.614	0.644
5	DFKI-SLT	0.7777	0.5151	0.6197
6	HSHL1	0.7216	0.5375	0.6161
7	Comtravo-DS	0.7042	0.5274	0.6031
8	LT-UHH	0.8496	0.3892	0.5339
9	NoTeam	0.4166	0.276	0.332
10	DexieDuo	0.0108	0.0034	0.0052

Table 6: Results of subtask B, best micro F-1 score by team

ing a good balance). We could reuse much of the work performed in other projects building a solid feature extraction and classification pipeline. We demonstrated the need for post-processing measures and how the traditional methods performed against new methods with this problem. Further, we improve a hierarchical classification open source library to be easily used in the multi-label setup achieving state-of-the-art performance with a simple implementation.

The high scoring of such traditional and light-weighted methods is an indication that this dataset has not enough amount (or variety) of data to use deep learning methods, so keyword-spotting/word-usage was already good whereas synonyms, context, negations, etc. were not so relevant. Nonetheless, the amount of such datasets will probably increase, enabling more deep learning methods to perform better.

Many small improvements were not performed, such as elimination of empty predictions and using

label names as features. This will be performed in future work.

6 Acknowledgements

We thank Mark Cieliebak and Pius von Däniken for the fruitful discussions. We also thank the organizers of the GermEval 2019 Task 1.

References

- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.
- Fernando Benites. 2017. *Multi-label Classification with Multiple Class Ontologies*. Ph.D. thesis, University of Konstanz, Konstanz.
- Fernando Benites and Mark Cieliebak. 2017. Hierarchical classification for news articles. In *SwissText 2017 : 2nd Swiss Text Analytics Conference*.

Fernando Benites, Pius von Däniken, and Mark Cieliebak. 2019. Twistbytes-identification of cuneiform languages and german dialects at vardial 2019. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 194–201.

Florian Brucker, Fernando Benites, and Elena Sapozhnikova. 2011. An empirical comparison of flat and hierarchical performance measures for multi-label classification with hierarchy extraction. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 579–589. Springer.

Tommi Jauhainen, Heidi Jauhainen, and Krister Lindén. 2018. HeLI-based experiments in Swiss German dialect identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 254–262.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397.

Eneldo Loza Mencía and Johannes Fürnkranz. 2010. Efficient multilabel classification algorithms for large-scale problems in the legal domain. In *Semantic Processing of Legal Texts*, pages 192–215.

Bruno C Paes, Alexandre Plastino, and Alex Alves Freitas. 2014. Exploring attribute selection in hierarchical classification.

Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artieres, George Palouras, Eric Gaussier, Ion Androultsopoulos, Massih-Reza Amini, and Patrick Galinari. 2015. Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2009. Classifier chains for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 254–269. Springer.

Carlos N Silla and Alex A Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72.

Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *Int J Data Warehousing and Mining*, 2007:1–13.

Y. Yang. 1999. An evaluation of statistical approaches to text categorization. *Information retrieval*, 1(1):69–90.

The HUIU Contribution to the GermEval 2019 Shared Task 1

**Melanie Andresen, Melitta Gillmann, Jowita Grala, Sarah Jablotschkin,
Lea Röseler, Eleonore Schmitt[†], Lena Schnee, Katharina Straka,
Michael Vauth, Sandra Kübler[‡], Heike Zinsmeister**

Universität Hamburg, [†]Universität Bamberg, [‡]Indiana University

{melanie.andresen, melitta.gillmann, sarah.jablotschkin}@uni-hamburg.de
 {Jowita.Grala, Lea.Roeseler}@studium.uni-hamburg.de
 {Lena.Schnee, Katharina.Straka}@studium.uni-hamburg.de
 {michael.vauth, heike.zinsmeister}@uni-hamburg.de
 eleonore.schmitt@uni-bamberg.de, skuebler@indiana.edu

Abstract

In this paper, we present the HUIU system for the GermEval 2019 shared task 1. Our system uses linear SVMs with word and POS unigrams and the number of authors as features. We obtain a micro-averaged F-score of 80.67 on the test data, thus ranking 15th out of 19 submissions, or 9th out of nine groups.

1 Introduction

This paper describes the contribution of the HUIU team to the shared task on hierarchical classification of book blurbs at GermEval 2019 (Reimus et al., 2019). The task is a multi-label classification that assigns categories to books. The labels constitute a hierarchy, i.e., there are several sub-labels to each category. Two tasks were offered: one focusing on assigning more general labels, the second one focusing on additionally assigning finer grained, hierarchical labels. Our team participated in the first task on assigning general labels, i.e., our system assigns each book one or more labels from the following set: 'Architektur & Garten' (Architecture & Gardening), 'Ganzheitliches Bewusstsein' (Holistic Awareness), 'Glaube & Ethik' (Belief & Ethics), 'Kinderbuch & Jugendbuch' (Books for Children and Young Adult Readers), 'Künste, Literatur & Unterhaltung' (Arts, Literature & Entertainment), 'Ratgeber' (Counseling), and 'Sachbuch' (Nonfiction).

The HUIU system was developed as a class project at the University of Hamburg, i.e., all authors participated in a 6-day compact course that provided an introduction to machine learning for linguists and digital humanities researchers, under the supervision of Kuebler and Zinsmeister. All participants had some experience in programming, but only one of the participants had had prior experience with machine learning. This project was

intended to provide a practical introduction to machine learning and to familiarize the participants with every step in the process of translating a problem into a machine learning problem, deciding on a machine learning algorithm, a feature set, extracting features, running machine learning experiments, and evaluating the outcomes. The team submitted a contribution to this shared task as well as to the GermEval 2019 shared task 2 (Andresen et al., 2019).

Because of the setting in a short compact course, the team decided to focus on standard machine learning algorithms available in scikit-learn (Pedregosa et al., 2011), with a fairly basic feature set and initially reducing the problem to a single label classification system. We then extended the feature set only minimally, and used a simple method to extend the classification approach towards a system where we can assign at most three labels. Also because of the course setting, we decided that we would not experiment with deep learning architectures.

The remainder of the paper is structured as follows: Section 2 discusses related work, section 3 describes our experimental setup, including the data set, the machine learning experiments, and the evaluation metrics. Section 4 shows the official results, and we discuss additional results on the development set: experiments to determine good settings for our thresholding approach to multi-label classification and a feature ablation study. We conclude in section 5 and discuss future work.

2 Related Work

Multi-label classification has not received much attention in the field of Computational Linguistics. The few exceptions concern work in the fields of offense detection (e.g. Ibromim and Budi, 2019), relation detection (e.g. Surdeanu et al., 2012), and

the prediction of medical codes in clinical notes ([Mullenbach et al., 2018](#)). These tasks are similar to our problem in that the number of labels differs per instance. For example, each tweet may contain abusive and/or hate speech and the latter may be related to one or to several issues such as creed, sexual orientation, or disability. Similarly, each sentence may contain a wide range of different relations, and each clinical note may contain a different number of medical codes. An interesting case is presented by [Chalkidis et al. \(2019\)](#), who have annotated legal texts with about 7 000 concepts from the European Vocabulary (EUROVOC). This does not only present a case of an extreme multi-label classification, but it also requires few-shot or one-shot learning approaches since most of these concepts are used very infrequently in the texts.

[El Kafrawy et al. \(2015\)](#) present an overview of methods for addressing multi-label classification and ranking. For multi-label classification, they distinguish between methods that transform the problem into single-label classification, adaptations of single-label classifiers, and ensemble methods. Problem transformations consist of sets of 1-vs-all classifiers, 1-vs-1 classifiers, or creating all combinations of labels and treating them as single labels. For classifier adaptation, neural networks are ideal since every label can be represented as a single output node, and depending on their activation level, multiple levels can be chosen, but other methods can be adapted as well. [El Kafrawy et al. \(2015\)](#) come to the conclusion that ensembles of classifiers work best in a multi-label classification situation.

3 Experimental Setup

3.1 Data Set

We use the data set provided by the shared task. It consists of a training set (containing 14 548 book blurbs), a development set (containing 2 079 book blurbs), and a test set (containing 4 157 book blurbs). For the final submission, we trained on the combination of the training plus the development set. For the additional experiments described in section [4.2](#), we trained on the training set and evaluated on the development set. Figure [1](#) shows an example of a book entry, reduced to the relevant fields.

Since we started with a single label classification system, we first used only the first label as-

```
<book date="2019-01-04" xml:lang="de">
<title>Die Essenz der Lehre Buddhas</title>
<body>Klar und verständlich führt der Dalai Lama in die buddhistische Lehre ein und eröffnet praktische Wege für alle, die Gelassenheit und inneren Frieden suchen. Wer diese einfachen, aber bewährten Grundsätze des Dalai Lama übernimmt und nach ihnen lebt, der lebt auch in Harmonie mit sich und seinen Mitmenschen dies ist die Essenz der Lehre Buddhas.</body>
<categories>
<category>
<topic d="0">Glaube & Ethik</topic>
</category>
<category>
<topic d="0">Ganzheitliches Bewusstsein</topic>
</category>
</categories>
<authors>Dalai Lama</authors>
<isbn>9783453702479</isbn>
</book>
```

Figure 1: Example of a book blurb.

signed to a book in the training data. However, the training data may contain more than one label per book. Therefore, we decided to add one training instance per label. I.e., a book with three labels would contribute three training instances, each being assigned one of the labels.

3.2 Extracted Features

We extracted word and part of speech (POS) n -grams as well as the number of authors as features. For the n -grams, we used the title and the body of the text, as delineated in the XML (see Figure [1](#) for an example). We then performed minimal tokenization via a script. For POS tagging, we used *TnT* ([Brants, 1998](#)), trained on the Tübingen Treebank of Written Language (TüBa-D/Z) ([Telljohann et al., 2006](#)), version 10.

For words and POS tags, we experimented with n -grams of length 1-3. In the final system, only unigrams were used as features since bigrams and trigrams negatively affected the results of the classifier. In addition to word and POS unigrams, we used the number of authors as a feature, using the number of commas as indicator of the number of authors.

3.3 Methodology

We used scikit-learn ([Pedregosa et al., 2011](#)) for our experiments. An initial investigation comparing SVMs (Support Vector Machines) and Random Forest classifiers showed that a linear

Rank	Team	Subset Acc.	Recall	Precision	micro-F
1	Ericsson Research	83.64	89.23	84.32	86.70
15	HUIU	75.63	80.63	80.72	80.67

Table 1: The official results of the HUIU system in comparison to the best performing system.

SVM gave the best results on the development set (in the single-label setting). For this reason, we only report experiments with the linear SVM. A non-exhaustive parameter search reached the best results using the default settings (penalty=l2, loss=squared_hinge, dual=True, tol=0.0001, C=1.0, multi_class=ovr, fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000).

3.4 From Single-Label to Multi-Class Classification

Since SVMs are inherently binary classifiers, they internally already split the problem into multiple classification steps. The linear SVM implementation in scikit-learn follows liblinear and implements a 1-vs-all strategy. We decided to use the internal results of the SVM by looking at the decision function provided for linear SVMs to decide whether we should add a second or third label. We used a manually determined threshold of the difference between the probability of the first and second label (or between the second and third respectively). Our best results are based on allowing a second label only and setting the threshold to ≤ 0.19 . For a closer look at the effects of setting thresholds and using multiple labels, see section 4.2.

3.5 Evaluation

For evaluation, we used the official scorer provided by the shared task. It reports precision, recall and the micro-averaged F-score, along with subset accuracy (i.e., the percentage of instances that were assigned the correct set of labels). The micro-averaged F-score serves as the main ranking function in the shared task.

4 Results

4.1 Official Shared Task Results

9 teams had submitted an overall number of 19 results. The HUIU contribution was ranked no. 15, or 9th group. Table 1 shows the HUIU official results in comparison to the best system. Our system is based on word and POS unigrams and the

number of authors as features, allowing up to two labels.

The results show that our results reach a micro-averaged F-score that is about 6 percent points lower than the best ranked system.

4.2 Additional Results

In this section, we report on additional experiments, where we evaluated on the development set. In an investigation the required number of labels, we use word and POS n -grams, but only create one instance per book in the training data, using the first label. In the ablation study, we start with the full system and then systematically take away options.

However, note that the ablation results need to be taken with a large grain of salt since different runs of the SVM with the same setting often result in larger differences than the differences between settings¹. The settings where we use one label per book seem to be stable, thus the experiments for determining the best number of labels are run only once per setting. The ablation experiments were run twice, and we report the averages. Ideally, every setting should be run several times, but the time constraints of this project did not allow such a procedure.

4.2.1 Number of Labels and Thresholds

Table 2 shows the results when we vary the number of permissible labels from 1 to 3, and it shows the effects of choosing corresponding thresholds. The threshold is defined as the difference between the probability the SVM assigns to the first and the second label (or the second and third respectively) in the internal 1-vs-all binary classifications. I.e., if we have a high threshold, corresponding to a large difference between the probabilities of the two labels, the system is very permissive in choosing a second label. If the threshold/difference is low, the first and second label need to be very close in probability for the second label to be added.

The results show that there are small differences in the F-score when allowing different numbers of

¹The cause for this large variation is unclear.

Setting	Threshold	Subset Acc.	Recall	Precision	micro-F
1 label	n/a	76.48	76.95	82.54	79.65
2 labels	1.0	68.40	83.50	74.84	78.93
	0.3	72.10	81.52	77.86	79.65
	0.2	73.64	80.40	79.20	79.80
	0.19	73.88	80.31	79.49	79.90
	0.15	73.93	80.09	79.48	79.79
	0.1	74.80	78.74	80.55	79.64
	0.05	75.52	77.80	81.34	79.53
3 labels	0.19; 0.2	73.79	80.81	78.62	79.70
	0.19; 0.19	73.79	80.85	78.70	79.76
	0.19; 0.18	73.79	80.85	78.73	79.78
	0.19; 0.17	73.79	80.76	78.78	79.76
	0.19; 0.15	73.79	80.76	78.85	79.80
	0.19; 0.12	73.88	80.63	78.96	79.79

Table 2: Results when varying the number of permissible labels and thresholds (on the development set).

Setting	Subset Acc.	Recall	Precision	micro-F
full version	74.73	81.17	80.20	80.68
no author	73.65	79.94	78.65	79.29
no POS	75.20	81.57	80.74	81.16
no author/POS	74.97	81.59	80.55	81.07
no author/POS/title	74.22	80.70	80.01	80.40
no author/POS/title; one instance	73.88	79.96	79.70	79.83
no author/POS/title; one instance/label	76.86	77.31	82.92	80.02

Table 3: Results of the ablation study (on the development set).

labels: When we use only one label, we reach an F-score of 79.65, the best result using two labels reaches 79.90, thus giving us a minor boost in performance. Surprisingly, subset accuracy is also highest when allowing only one label. Allowing a third label results in an optimal F-Score (for this setting) of 79.80, i.e., it does not reach the highest F-score when using two labels.

However, when we look at the precision and recall scores, we see a different picture: Using one label gives a high precision but rather low recall, which is understandable since all books that have more than one label in the gold standard will at best be classified only partially. However, this setting also reaches the highest subset accuracy. Adding a second label with a high threshold of 1.0 reverses this picture, i.e., we gain in recall by adding more labels, but precision suffers. The more we lower the threshold the more we lose in recall but gain in precision. Thus, we need to find a good balance for the threshold.

4.2.2 Ablation Study

Table 3 shows the results of our ablation experiments. We start with the full system that also served as the basis for the official submission. We see that leaving out the number of authors results in a minor deterioration, but leaving out the POS information results in a boost in F of about 0.4, equally distributed across precision and recall. We had originally decided to use POS unigrams since they improved results in the single-label setting. This shows that the ideal settings do not transport across single-label and multi-label experiments.

Leaving out both author and POS information results in a minimal loss, leaving out the title information and using only one instance per book with the first label result in a smaller loss. Restricting the system to a single-label task results in a minimal improvement in F, based on high precision, but low recall. Surprisingly, this setting also provides the highest score for subset accuracy.

5 Conclusion and Future Work

This project was mostly carried out in the setting of a 6-day compact course. Given the time constraint, we have shown that we can put together a fairly robust system for multi-class classification of books into categories. Our system ranked about 6 points below the best performing system.

Future work should investigate using additional features, such as looking into sentence length, the syntactic complexity of sentences, or the occurrence of named entities. We also need to investigate the issue of variation in the SVM results when we use more than one instance per book while there is no variation at all when we only use one instance. Another point is to investigate ensembles as suggested by El Kafrawy et al. (2015).

References

- Melanie Andresen, Melitta Gillmann, Jowita Grala, Sarah Jablotschkin, Lea Röseler, Eleonore Schmitt, Lena Schnee, Katharina Straka, Michael Vauth, Sandra Kübler, and Heike Zinsmeister. 2019. The HUIU contribution to the GermEval 2019 shared task 2. In *Proceedings of the GermEval 2019 Workshop*, Erlangen, Germany.
- Thorsten Brants. 1998. *TnT-A Statistical Part-of-Speech Tagger*. Universität des Saarlandes, Computational Linguistics, Saarbrücken, Germany.
- Ilias Chalkidis, Emmanouil Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2019. Extreme multi-label legal text classification: A case study in EU legislation. In *Proceedings of the Natural Legal Language Processing Workshop 2019*, pages 78–87, Minneapolis, MN.
- Passent El Kafrawy, Amr Mausad, and Heba Esmail. 2015. Experimental comparison of methods for multi-label classification in different application domains. *International Journal of Computer Applications*, 114(19).
- Muhammad Okky Ibrohim and Indra Budi. 2019. Multi-label hate speech and abusive language detection in Indonesian twitter. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 46–57, Florence, Italy.
- James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the ACL*, pages 1101–1111, New Orleans, LA.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Steffen Remus, Rami Aly, and Chris Biemann. 2019. GermEval-2019 task 1: Shared task on hierarchical classification of blurbs. In *Proceedings of the GermEval 2019 Workshop*, Erlangen, Germany.
- Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465, Jeju Island, Korea.
- Heike Telljohann, Erhard W. Hinrichs, Sandra Kübler, and Heike Zinsmeister. 2006. *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*. Seminar für Sprachwissenschaft, Universität Tübingen, Germany.

Label Frequency Transformation for Multi-Label Multi-Class Text Classification

Raghavan A K

Global AI Accelerator, Ericsson / Chennai
k.raghavan.a@ericsson.com

Venkatesh Umaashankar

Ericsson Research / Chennai
venkatesh.u@ericsson.com

Gautham Krishna Gudur

Global AI Accelerator, Ericsson / Chennai
gautham.krishna.gudur@ericsson.com

Abstract

In this paper, we (**Team Raghavan**) describe the system of our submission for GermEval 2019 Task 1 - *Subtask (a)* and *Subtask (b)*, which are multi-label multi-class classification tasks. The goal is to classify short texts describing German books into one or multiple classes, 8 generic categories for *Subtask (a)* and 343 specific categories for *Subtask (b)*. Our system comprises of three stages. **(a)** Transform multi-label multi-class problem into single-label multi-class problem. Build a category model. **(b)** Build a class count model to predict the number of classes a given input belongs to. **(c)** Transform single-label problem into multi-label problem back again by selecting the top- k predictions from the category model, with the optimal k value predicted from the class count model. Our approach utilizes a *Support Vector Classification* model on the extracted vectorized *tf-idf* features by leveraging the *Byte-Pair Token Encoding (BPE)*, and reaches f1-micro scores of **0.857** in the test evaluation phase and **0.878** in post evaluation phase for *Subtask (a)*, while **0.395** in post evaluation phase for *Subtask (b)* of the competition. We have provided our solution code in the following link: <https://github.com/oneraghavan/germeval-2019>.

1 Introduction

Multi-label Multi-class Hierarchical Classification tasks typically encompass multiple possible (one or more) labels for each instance (not mutually exclusive) across multiple possible classes (two or more) with many levels of hierarchies, and are widely used in domains like text classification (Rousu et al., 2006), image classification (Hsu et al., 2009) and bioinformatics (Barutcuoglu et al., 2006), (Feng et al., 2017). In this paper, we present our submission approach for *Subtask (a)*

and *Subtask (b)* in *GermEval 2019 Task 1* (Remus et al., 2019). Here, we convert our task into two sub-problems: first, to predict the category; second, to predict the class frequency corresponding to the multi-label setting. Our approach can be broadly split into three stages.

- Transform the multi-label multi-class problem into a single-label multi-class problem, and build a category model.
- Build a class count predictor model to predict the number of classes that a given input could be categorized.
- Transform single-label problem back into a multi-label problem by selecting the top k predictions from the category model, with the optimal k value predicted from the class count model.

Conventionally, Natural Language Processing (NLP), particularly text classification tasks have been modeled using variants of Support Vector Machines (Joachims, 1998) and Naive Bayes Classifiers (McCallum et al., 1998). With the widespread adoption of deep learning models, there has been considerable increase in efficiencies for such tasks, however they are still considered black box models, and it is extremely hard to interpret them, in contrast to conventional Machine Learning algorithms. Moreover, the time and computational resources required for training deep neural networks are extremely higher than conventional Machine Learning models.

Hence, in our approach, we utilize the traditional *Support Vector Classification* modeled using *tf-idf* (term frequency - inverse document frequency) feature vectors combined with class count predictor, and we leverage the *Byte-Pair Encoding (BPE)* compression tokenization mechanism. Experimental results from exploiting

such simple model fusion approaches show that in the post-evaluation phase, f1-micro scores of 0.878 on *Subtask (a)* and 0.395 on *textitSubtask (b)* could be achieved. The overall modeling pipeline/architecture of our approach can be found in Figure 1.

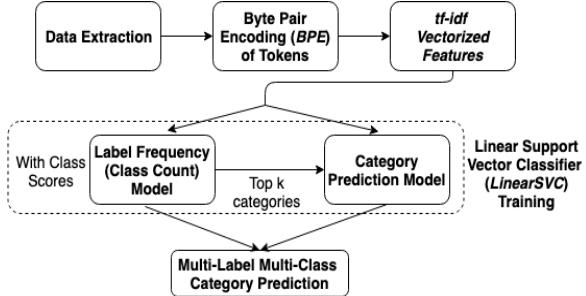


Figure 1: Modeling Pipeline/Architecture for both *Subtask (a)* and *Subtask (b)*

The rest of the paper is organized as follows. The characteristics of the dataset are described in Section 2. In Section 3, the data extraction and pre-processing techniques to obtain our feature vectors are discussed. Section 4 presents our model architecture along with training and aggregation phases. This is followed by systematic evaluation of our model’s results and submission in Sections 5 and 6, and we finally conclude the paper.

2 Data Description

The *GermEval 2019 Task 1* dataset (Remus et al., 2019) consists of German books crawled from randomhouse.de, with the following attributes – title, description, author name, ISBN and book release date. Apart from date, most of the other features available are in the form of text, where title and description are very short texts. A total of 343 categories are present across three levels of hierarchy with 8, 93 and 242 categories in each level, and multiple labels can be assigned to each book.

The corpus has a very imbalanced label distribution. Figure 2 shows the top-level label distributions, while Figure 3 shows the top 30 label distributions, and it can be vividly inferred that the label categories are highly skewed.

3 Data Extraction and Prepossessing

The corpus was presented as an XML file, with XML tags for each feature. The XML files were parsed using the Python library - *BeautifulSoup*,

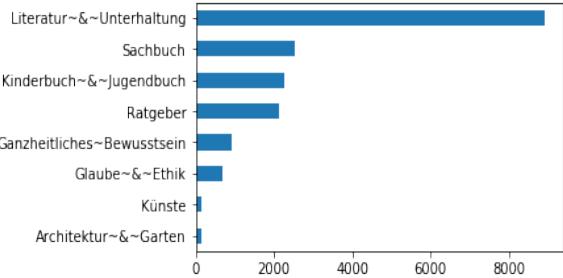


Figure 2: Top-level Label Distribution

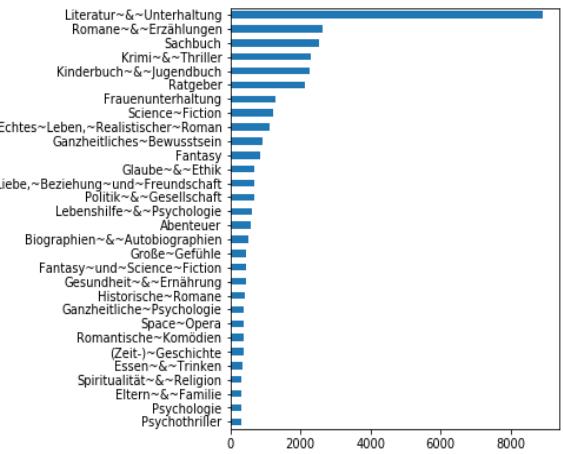


Figure 3: Label Distribution for top 30 classes with hierarchies

and the columns given in the corpus such as title, description, list of authors, date of publication, ISBN were extracted from the XML format into a CSV format. The corpus has hierarchy information in the *hierarchy.txt* file, which was used to validate and prepare the corpus for modelling.

From the extracted data, we initially filter out the stop words and numbers from title and description for every book using the *Natural Language Toolkit (NLTK)* library available in Python. We then utilize **Byte-Pair Encoding (BPE)** (Heinzerling and Strube, 2018) based tokenization to create tokens from the titles and descriptions for various texts. BPE tokenization, in this context, identifies the most common consecutive bytes of German words and replaces with a byte that does not occur within the data. For instance, ‘*Ein Blick hinter die*’ might be converted into ‘ein blick hinter die’ after BPE with a vocabulary size of 25,000. We utilize the BPEmb library for the same, which utilizes pre-trained byte-pair embeddings, and require no tokenization, also being much shorter.

The German words are split into multiple smaller sub-words, which would inherently en-

able better representations for each book’s Title and Description, and provide more context for the feature vectors which are learned. Subwords in BPEmb also enable effective guessing of unknown/out-of-vocabulary words’ meaning based on context. For instance, the suffix -shire in Melfordshire indicates a location.

After tokenizing each title and description of books using *BPE*, we create ***term frequency - inverse document frequency (tf-idf)*** vectors from the tokens. *tf-idf* is a widely used statistical measure in domains of NLP, text mining and information retrieval, which signifies the importance of a word to a document in the whole corpus (Ramos, 2003).

We experimented with various contiguous sequences of n-grams – unigrams, bigrams and trigrams for creating *tf-idf* vectors, and observed that bigram models yielded better results than the rest. Each book can have multiple authors, so we treat the authors’ data as a binomial attribute (creating a *Label Binarizer* which returns 1 if the book was authored by that author, else 0) across all authors. We extract the *year* from the publication date, and utilized it as a categorical feature. We also extracted the ***Group ID*** and ***Publisher ID*** from the 13-digit ISBN and represented them as categorical features again.

We created two sets of target variables from the extracted data – one with the categories as target labels, and other with the count of categories. Hence, we also created and pipelined two different models – *class count model* and *category score predictor model*. For instance, for a given book instance, if there are k categories, then that training sample has been duplicated k times with each sample having one category. Adding all features, a sparse feature matrix was created. The final feature matrix is of size (17783 x 594931) for both class count model and category score predictor model.

Labels	Counts
Label 1	15549
Label 2	1004
Label 3	70
Label 4	4

Table 1: Class Count Frequencies

The class count model is a classification problem with labels 1, 2, 3 as class frequencies. While

there are book instances up-to 4 top-level observed class frequencies, there are only 4 training samples for category 4 which is extremely less (negligible), hence we consider only 3 categories. The class count distribution can be observed in Table 1. The categories are predicted based on an 8-class classification approach for top-level categories (*Subtask (a)*), and a 343-class classification mechanism for multi-level (*Subtask (b)*) categories.

4 Model Pipeline

Given the corpus has a heavy class imbalance across all levels, we choose a Support Vector Machine (SVM) based model for the task. SVMs are known to exhibit robustness and perform effectively under class imbalance (Tang et al., 2009).

The ***Linear Support Vector Classifier (LinearSVC)*** in a multi-class classification problem utilizes a one vs rest scheme, wherein the objective is to return the best-fit hyperplane that categorizes the data in an n-dimensional space. Identifying the right hyperplane is primarily dependent on the *margin* (maximized distance between hyperplane and nearest data point), and the loss function governing the margin. We utilize the *squared-hinge loss* for the same, as it is widely used for maximum-margin classification in SVMs that penalizes the violated margins more strongly (quadratically). The *squared-hinge loss*, l for Linear SVM is given by,

$$l(y) = \max(0, 1 - t \cdot y)^2$$

where y is the classifier score $y = w \cdot x + b$, w, b are the parameters of the hyperplane, x is the input variable(s) and the intended output $t = \pm 1$.

We created two Linear SVC models, one for predicting the count of classes a book could belong to, and the other for category score predictor. *Linear SVC* was chosen as it uses the *liblinear* framework and scales well with the increase in the number of features (Fan et al., 2008). Moreover, it is computationally faster than most other Linear SVM implementations, and utilizes many other advanced optimization techniques. We utilize the *scikit-learn* Python framework (Pedregosa et al., 2011) to train and test the *LinearSVC* model, which uses liblinear as its default implementation. Also, this implementation offers flexibility in the choice of penalties and loss function parameters, and would inherently scale well to larger samples of data.

In our case, use of *tf-idf* vectors to represent words created a large number of feature vectors. The *tf-idf* vector representation were very sparse owing to the short text representations for each book. Compressed sparse representation (*CSR*) of the feature matrix was further utilized reduce the size of the training set.

4.1 Model Parameters

After extensive parametric optimization for both models – count classifier and category classification using grid search, we arrived the following set of final parameters to yield best efficiencies. The optimal parameters for the LinearSVC model are elucidated in Table 2.

Parameter	Optimal Value
C	1.0
Tolerance for stopping	0.0001
Loss	Squared Hinge Loss
Penalty	L2
Optimization algorithm	Dual
Max Iterations	3000

Table 2: Optimal Parameters for *LinearSVC*

For the top-level classifier, the following class weights are also used to handle class imbalance. We utilized grid search to fine tune the class weights again, which can be observed in Table 3.

Category	Class Weight
Kinderbuch & Jugendbuch	1.8
Ratgeber	3
Sachbuch	2
Glaube & Ethik	2
Künste	6
Architektur & Garten	6
Literatur & Unterhaltung	1
Ganzheitliches Bewusstsein	1

Table 3: Class Weights for Top-level Categories to handle Class Imbalance in *LinearSVC*

The class weights might be conventionally perceived that the categories with lower cardinality might have higher class weights. However, the model prioritizes and takes into account the confusion between various classes than the imbalance in classes alone. For instance, we could observe that a lot of Sachbuch and Glaube get classified as Literatur & Unterhaltung, hence giving more weightage to the latter aids in higher efficiencies.

4.2 Model Fusion

We pipeline the class score predictor and class count predictor models together for effective classification of categories. Since the input features for both models remain the same, we train our model with categories as target variables for the class score predictor model, and with class counts as target variables for class count predictor model.

In the class count model, we also add the scores of output class predictions to the input feature space. This is motivated by the inherent fact that there is a high correlation between the number of categories a book belongs to, and its corresponding class with the highest score. While predicting a book’s category, we first get the class scores for all categories from the class score predictor model, and then append those predictions with input data to the class count classifier model. Once the class count predicts number of possible categories k , we find the top k category predictions from the class category predictor (likelihood) model.

In this way, the class imbalanced multi-label problem is split into two simple prediction problems. Also, by splitting the problem into smaller chunks, we are able to train multiple models in parallel, thus reducing the total training time of the system. With the above setup, retraining the entire set of models takes just under 2 minutes.

5 Evaluation

For building an end-to-end classifier system, we build a Classifier Class extending the *scikit-learn* Base estimator API, with the respective fit and predict functions. The constructor parameters passed, are the hyperparameters for the class count predictor and class likelihood predictor models. Inside the constructor, we create two LinearSVC models. In the fit (training) phase, both the predictors are trained with their respective target variables. We utilize a ***K-Fold Cross-validation*** strategy ($K = 4$) and get the class scores to be used in training for class count predictor. Once we have the class scores, we retrain the class predictor with the whole training data again. The class count predictor is then utilized for training along with the class scores appended.

Similarly, in the predict phase, the class prediction model is first used to obtain class prediction scores, and further utilized by class count predictor to get the class count distribution. We select take k highest category predictions from the class

scores.

The micro-f1 scores for *Subtask (a)* and *Subtask (b)* with CV=4 folds can be found in Table 4.

CV Fold	<i>Subtask (a)</i>	<i>Subtask (b)</i>
Fold 1	0.833	0.384
Fold 2	0.943	0.471
Fold 3	0.950	0.484
Fold 4	0.900	0.397

Table 4: k-fold Cross-Validation (k=4) on *Subtask a* and *Subtask b*

The experimental setup (HW/SW configurations) utilized for our solution are as follows:

(1) Intel® Xeon® Processor E5-2650 v4 30M Cache, 2.20 GHz, 12 Cores, 24 Threads (2) 250 GB RAM (3) CentOS 7.

6 Submission and Results

With the above setup, the model was able to achieve the results showcased in Table 5 in the test phase.

Phase	<i>Subtask (a)</i>	<i>Subtask (b)</i>
Validation Phase	0.851	0.4098
Test Phase	0.857	-
Post Evaluation Phase	0.878	0.3947

Table 5: Evaluation Metrics (f1-micro scores) for Test Data on *Subtask a* and *Subtask b*

Team Raghavan achieves rank 4 in *Subtask (a)* during the test phase (f1-score of ~ 0.86). However, in the post-evaluation phase, we achieve an f1-score of 0.878, which secures us the first position in *Subtask (a)*. The additional 0.02 gain in micro-f1 score during post-evaluation phase finally was achieved by adding ISBN based features – *Group ID* and *Publisher ID*.

7 Conclusion

In this paper, we have successfully demonstrated that traditional approaches like Linear Support Vector Machine Classifier, with a class count predictor model can effectively model Multi-label Multi-class Hierarchical Text Classification of German blurbs – *GermEval Task 1*. The model designed by us was aimed for top-level categories, i.e., *Subtask (a)*, which implements flat classification and doesn't make use of much hierarchical

dependencies. That is one primary reason for *Subtask (b)* achieving relatively less efficiencies.

The authors would like to emphasize that conventional machine learning solutions would help in better interpretability, and when pipelined/fused with the right set of techniques, can effectively save a lot computational resources and time.

8 Credits

The authors would like to thank their colleagues at Ericsson Research and Global AI Accelerator (GAIA) at Ericsson during this competition. Also, we would like to thank the *scikit-learn* developers for actively developing and maintaining this awesome tool. Finally, we thank the *GermEval* competition organizers for fostering a friendly and collaborative environment around this dataset and for answering our questions throughout the competition.

References

- Zafer Barutcuoglu, Robert E Schapire, and Olga G Troyanskaya. 2006. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836.
- Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.
- Shou Feng, Ping Fu, and Wenbin Zheng. 2017. A hierarchical multi-label classification algorithm for gene function prediction. *Algorithms*, 10(4):138.
- Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA).
- Daniel J Hsu, Sham M Kakade, John Langford, and Tong Zhang. 2009. Multi-label prediction via compressed sensing. In *Advances in neural information processing systems*, pages 772–780.
- Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.
- Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, pages 41–48. Citeseer.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Pas-sos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Juan Ramos. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning*, volume 242, pages 133–142.

Steffen Remus, Rami Aly, and Chris Biemann. 2019. GermEval-2019 task 1: Shared task on hierarchical classification of blurbs. In *Proceedings of the GermEval 2019 Workshop*. Erlangen, Germany.

Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. 2006. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7(Jul):1601–1626.

Y. Tang, Y. Zhang, N. V. Chawla, and S. Krasser. 2009. Svms modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):281–288.

Logistic Regression and Naive Bayes for Hierarchical Multi-label Classification at GermEval 2019 - Task 1

Kristian Rother and Achim Rettberg

Hochschule Hamm-Lippstadt

Marker Allee 76-78

59063 Hamm, Germany

kristian.rother@hshl.de

achim.rettberg@hshl.de

Abstract

This paper describes an entry of two systems for Task 1 of GermEval 2019. Task 1 consists of classifying labels of genres for German books based on short advertisement text blurbs. It is split into Subtask A, a multi-label classification task with 8 classes and Subtask B a hierarchical multi-label classification task with 343 different classes. The submitted systems were used for both subtasks and combined Logistic Regression and Naive Bayes to build a classifier. To reach the final systems, different models and combinations of hyperparameters were explored empirically. The best submitted system reached micro-averaged F1-scores of 0.82 and 0.62 for the two subtasks respectively.

1 Introduction

Hierarchical multi-label classification (HMC) of blurbs is the task of classifying multiple labels for a short descriptive text, where each label is part of an underlying hierarchy of categories. The increasing amount of available digital documents and the need for more and finer grained categories calls for new, more robust and sophisticated text classification methods. Large datasets often incorporate a hierarchy which can be used to categorize information of documents on different levels of specificity. The traditional multi-class text classification approach is thoroughly researched, however, with the increase of available data and the necessity of more specific hierarchies and since traditional approaches fail to generalize adequately, the need for more robust and sophisticated classification methods increases ([Remus et al., 2019](#)).

To advance the state of the art in HMC, Task 1 of GermEval 2019 was launched. Task 1 consists of classifying labels of genres for German books based on short advertisement text blurbs. It is split into Subtask A, a multi-label classification task

with 8 classes and Subtask B a hierarchical multi-label classification task with 343 different classes. For Subtask B, each label is part of an underlying hierarchy of categories. This subtask is thus a Hierarchical multi-label classification (HMC) problem.

HMC problems exist in a variety of domains from text classification tasks to the prediction of gene functions. The text classification research is mostly focused on problems in English. Typically, (deep) neural models are applied ([Liu et al., 2017](#); [Gargiulo et al., 2019](#); [Shimura et al., 2018](#); [Cerri et al., 2014](#)). But there are also other approaches like decision trees ([Vens et al., 2008](#)) or genetic algorithms ([Cerri et al., 2012](#); [Gonçalves et al., 2018](#)). An overview of multi-label classification algorithms can be found in ([Sharma and Mehrotra, 2018](#)).

To compete in GermEval 2019 and to contribute to the state of the art in German HMC, two systems, that only vary in the n-grams that they used, were submitted:

- *HSHL_LogisticRegression_NaiveBayes1*
(from here on out referred to as System 1)
- *HSHL_LogisticRegression_NaiveBayes2*
(from here on out referred to as System 2)

Apart from participating in GermEval 2019, the intended use case for the submitted systems is to serve as an introductory tool for machine learning in short talks or lectures and as a baseline for more complex systems. As such, some additional constraints were imposed on the systems that go beyond the scope of the GermEval task:

- End-to-end runtime for Subtask A around 10 minutes.
- End-to-end runtime for Subtask B around 45 minutes.

- No use of additional data.
- One system, no ensemble.

2 Data

The dataset (BGC-DE) was crawled from Random House and provided by the organizers. It was split into 70% training data, 10% validation data and 20% test data. The labels for the test data were not given to the participants because this data served as the means to judge the submissions. The dataset contains information on German books in XML-format.

Apart from the advertisement text blurbs and genres, some additional metadata (title, author, URL, date of publication) and the ISBN of the book were provided. Some quantitative characteristics of the dataset are summarized in table 1.

Number of samples	20,784
Average length of blurb	94.67
Total number of classes	343 (8, 93, 242)

Table 1: Characteristics of the BGC-DE dataset. The 343 classes form a hierarchy of 8 classes at the first level, 93 classes at the second level and 242 classes at the third level.

After an initial preparation phase in which a tiny sample of data was provided to check the file format, the data was released in two phases. In phase one, a validation set was released which consisted of labeled training data and corresponding unlabeled test data. After phase one ended, the labels for the training data of the validation set were released and the validation set essentially became the new training set for phase two with new unlabeled training data provided for this phase.

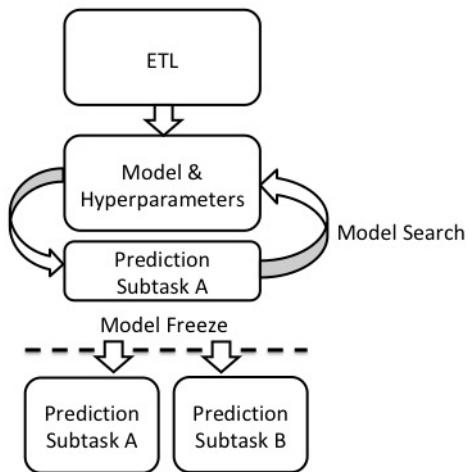
3 Experimental Setup

To produce a submission for the task, the provided data was first loaded, sanitized and converted to a format that is suitable for the use in machine learning, namely a pandas DataFrame (ETL). Because some entries had no blurb, the book title was used as a filler value. Additionally the genre names were sanitized by lowercasing them and removing special characters and spaces so they could be used as labels for the DataFrame. No additional outside data was used.

To find a model and suitable hyperparameters, the provided training data from phase one of

the competition was split into 80% training data and 20% development data. Afterwards, different models and hyperparameters were trained and evaluated for Subtask A on this new split (Model Search). This train-evaluate cycle was repeated iteratively until no further improvements could be made (Model Freeze). The chosen model was used to make the final predictions for both Subtask A and Subtask B. The overall process is shown in figure 1.

Figure 1: Overall process for the competition.



3.1 Technical Resources

All experiments were conducted in Jupyter Notebooks, version 4.0.2 ([Kluyver et al., 2016](#)) running a Python 3.5.0 ([Python Software Foundation, 2019](#)) kernel with the following libraries:

- pandas 0.23.4 ([McKinney, 2010](#))
- NumPy 1.11.3 ([Oliphant, 2006](#))
- scikit-learn 0.20 ([Pedregosa et al., 2011](#))
- spaCy 2.0.12 ([Honnibal and Montani, 2019](#))

A fixed seed was used for the random number generators. All models were trained on an end of 2013 MacBook Pro with a 2 GHz Intel Core i7, 8 GB 1600 MHz DDR3 and an Intel Iris Pro 1536 MB GPU¹.

4 Model Search

In order to find a model and suitable hyperparameters, seven different models from the scikit-learn

¹With this setup, the entire end-to-end training on the provided development set took 8:24 minutes for Subtask A and 46:55 minutes for Subtask B.

library were trained and evaluated on Subtask A. The selected models were Decision Tree, Random Forest, Multinomial Naive Bayes, K-nearest Neighbors (KNN), SVC, Linear SVC and Logistic Regression. To get a baseline for each model, the default values of the library were used. Afterwards, each model was optimized until no further improvements could be made. All parameters were found empirically by trying different combinations². The chosen hyperparameters of the optimized models are summarized in the following chapters with a list of the used parameters in Python Code for each model.

4.1 Decision Tree

For the decision tree, a random splitter was used and the minimum number of samples was set to 15. The default "Gini-criterion" yielded better results than the "Entropy-criterion". Balanced class weights or changing the number of features for the split to anything other than the total number of features did not improve the results.

- splitter='random'
- min_samples_split=15

4.2 Random Forest

For the random forest, 200 estimators were used and the minimum number of samples required to split an internal node was set to 5. Additionally, no bootstrapping was used and balanced subsamples were used for the class weights.

- n_estimators=200
- min_samples_split=5
- bootstrap=False
- class_weight='balanced_subsample'

4.3 Multinomial Naive Bayes

For Multinomial Naive Bayes, only the alpha value for smoothing was tuned. An alpha value of 0.08 yielded the best results. The default of learning the class prior probabilities outperformed using uniform priors.

- alpha=0.08

²The exact experiments that were conducted can be found in the lab notes that accompany the code at <https://github.com/rother/germeval2019>

4.4 K-nearest Neighbors

For KNN, a total of 9 neighbors were used. Weighting points by the inverse of their distance instead of weighting them equally (uniform) provided the best results.

- weights='distance'
- n_neighbors=9

4.5 Support Vector Machine - SVC

For the SVC, tuning the C value for regularization proved most useful. Ultimately, it was set to 15,900. Furthermore, balanced class weights were used. No other experiments yielded improvements.

- C=15900.0
- class_weight='balanced'

4.6 Support Vector Machine - Linear SVC

Due to technical problems³ the Linear SVC had to be constructed by passing kernel='linear' and probability=True to SVC instead of using LinearSVC directly. The only optimization that was performed was using balanced class weights, which improved the overall results.

- kernel='linear'
- probability=True
- class_weight='balanced'

4.7 Logistic Regression

For the logistic regression, a liblinear solver with a maximum number of 1000 iterations, automatic multiclass fitting and balanced class weights was used. L2 regularization with C=40.0 was applied. No dual formulation was used as the number of samples was bigger than the number of features.

- C=40.0
- dual=False
- multi_class='auto'
- max_iter=1000
- class_weight='balanced'

³Namely, that LinearSVC does not provide a predict_probab() method.

5 Results of the Model Search experiments

The micro-averaged F1-scores for all optimized models when evaluated on Subtask A are summarized in table 2. The table is ordered in ascending order of F1-scores. The Logistic Regression model performed best and was thus picked for the competition.

#	Model	Default	Optimized
1	Decision Tree	0.6049	0.6125
2	Random Forest	0.6125	0.6667
3	Naive Bayes	0.6253	0.7703
4	KNN	0.7164	0.7377
5	SVC	0.5127	0.7878
6	Linear SVC	0.7731	0.7882
7	Logistic Regression	0.6919	0.7883

Table 2: Overview of micro-averaged F1-scores for the default configuration of models and optimized versions. Bold indicates the best model.

Additionally, one mixed soft-voting ensemble that combined all models and ensembles that combined the two or three best models were created⁴. The ensembles that used the top models improved upon the best F-score as summarized in table 3. However, they were not used for the submission as explained in the introduction.

#	Ensemble Name	Models	F-score
1	All	1, 2, 3, 4, 7	0.7710
2	Top 2b	3, 7	0.7967
3	Top 3	3, 4, 7	0.7950
4	Top 2a	4, 7	0.8001

Table 3: Overview of ensembles with micro-averaged F1-scores. Bold indicates improvements over the best single model.

6 Model Freeze

Because the Logistic Regression model yielded the best results during Model Search, it was used for the submissions. Both submissions combined this Logistic Regression model with a Naive Bayes approach similar to (Wang and Manning, 2012)⁵ to classify the genres of German books from ad-

⁴Linear SVC and SVC were not included in these ensembles due to technical problems.

⁵See also <https://www.kaggle.com/jhoward/nb-svm-strong-linear-baseline>

vertisement text blurbs. spaCy was used as the tokenizer and unicode accents were stripped but the casing was kept for both submissions. Both lemmatization and stopwords lowered the results and thus were not used. Empty blurbs were replaced with the title of the book. The tokenization parameters are summarized in table 4.

Parameter	Value
Tokenizer	spaCy (German)
Accent Stripping	Unicode
Lowercase	No
Lemmatization	No
Stopwords	No
Replace-Empty Blurbs with	Title

Table 4: Tokenization parameters.

For the term-frequency matrices, only words that appeared in at least 4 documents were used and words that appeared in more than 40% of documents were ignored. Inverse document-frequency-reweighting, sublinear term frequency scaling and smoothing were applied. The parameters are summarized in table 5. System 1 and System 2 only differ in the n-grams that were used to construct the term-frequency matrices. For System 1, only unigrams were used and for System 2, bigrams were used.

Parameter	Value
Minimum Number of Documents	4
Maximum Frequency of Documents	40%
Inverse document-frequency-reweighting	Yes
Sublinear Term Frequency Scaling	Yes
Smoothing	Yes

Table 5: Parameters for the term-frequency matrices.

For the logistic regression, the model from the Model Search stage was used. The hyperparameters are summarized in table 6.

Parameter	Value
Solver	liblinear
Maximum Iterations	1000
Multiclass	Automatic
Class Weights	Balanced
Dual Formulation	No
Regularization	L2
C	40.0

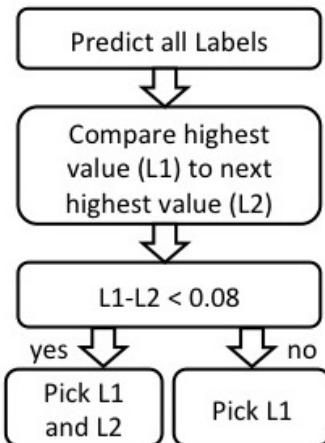
Table 6: Parameters for the logistic regression.

7 Classification procedure

For Subtask A up to two labels were classified per blurb, because using more than two labels lead to worse results. For the second label a limit of 0.08 was found to be optimal. This limit means that a second label is only classified, if the difference between the probabilities of the top two predicted labels is less than this value. This is depicted in figure 2.

For Subtask B, the system started with the classified level 1 labels from Subtask A. Based on these labels it moved up one level on the hierarchy and added exactly one additional level 2 label per level 1 label if a cutoff value was met. Afterwards, the system once again moved up a level on the hierarchy and added level 3 classifications. This process is depicted as Algorithm 1. The exact procedure is explained in more detail below.

Figure 2: Subtask A classification process.



On the second level, a list of all level 2 labels that follow the classified level 1 labels on the hierarchy was generated. The level 2 label with the highest probability was picked and added as a classification if the probability was higher than a provided cutoff value. Note that on level 2, only upto one label per level 1 label was picked.

For the third level, the level 2 classifications were used as a basis. If there was a classified level 2 label, a list of all level 3 labels that follow the classified level 2 label on the hierarchy was generated. If there was only one level 2 label (and thus also only one level 1 label) the system added as many level 3 labels as met the provided multi-cutoff for level 3. If there was a second level 2 label (and thus also a second level 1 label) a single level 3 label was added, if it met the provided cut-

Algorithm 1 Subtask B classification process.

```

Level1Labels ← SubtaskA
for all Level1Labels do
    Level2Probabilities ← get_from_l1()
    max_l2 ← Level2Probabilities(0)
    if max_l2 > 0.09 then
        Level2Labels ← get_label_l2()
    end if
end for
if len(Level2Labels) ≠ empty then
    ProbA ← get_from_l2(Level2Labels(0))
    ProbB ← get_from_l2(Level2Labels(1))
    for all ProbA do
        if probability > 0.15 then
            Level3Labels.append(get_label_l3())
        end if
    end for
    if ProbB ≠ null then
        max_l3 ← ProbB(0)
        if max_l3 > 0.7 then
            Level3Labels.append(get_label_l3())
        end if
    end if
end if
  
```

off value. This ensured that the lower confidence prediction from the earlier level can get at most one additional level 3 label.

The cutoff values were found empirically and are summarized in table 7. On the third level, a multi-cutoff value of 0.15 was used for the first level 2 label and a cutoff of 0.7 was used if there was a second level 2 label.

Level	Cutoff Value	Number of Labels
2	0.09	0 to 1 per level 1 label
3	0.15 / 0.7	0 to n per level 2 label

Table 7: Cutoff values and maximum number of labels for the classification procedure.

8 Results

Unigrams (System 1) performed slightly better than bigrams (System 2). Both systems participated in Subtask A and Subtask B of the competition. The final results are summarized in table 8 with the best results highlighted in bold.

System	F1-score A	F1-score B
System 1	0.8201	0.6161
System 2	0.8163	0.6063

Table 8: Final results. F1-scores are micro-averaged.

9 Conclusion

This paper presented two submission (System 1 and System 2) that were entered for Task 1 at GermEval 2019.

Both submissions use a combined Logistic Regression/Naive Bayes approach to classify genres of German books from advertisement text blurbs. spaCy was used as the tokenizer and unicode accents were stripped but the casing was kept for both submissions. The first submission uses unigrams and the second submission uses bigrams. The other hyperparameters are the same.

For the term-frequency matrices, only words that appeared in at least 4 documents were used and words that appeared in more than 40% of documents were ignored. Inverse document-frequency-reweighting, sublinear term frequency scaling and smoothing were applied.

For the logistic regression, a liblinear solver with a maximum number of 1000 iterations, automatic multiclass fitting and balanced class weights was used. L2 regularization with C=40.0 was applied. No dual formulation was used as the number of samples was bigger than the number of features.

The cutoff values for the classification procedure are summarized in table 7.

To facilitate the reproduction of the results, all code is made available as a Jupyter Notebook at one of the authors Github repositories⁶. Additional lab notes on the conducted experiments will also be made available at the same repository.

The end-to-end execution time on a consumer grade laptop was 8:24 minutes for Subtask A and 46:55 minutes for Subtask B and can be reduced further by storing and reloading intermediate results.

The unigram version (System 1) performed best and reached micro-averaged F1-scores of 0.82 and 0.62 for the two subtasks respectively. When only the best entry for each teams is counted, this means compared to the other participants rank 7 of 9 was reached for Subtask A and rank 5 of 6 was

reached for Subtask B. The winning scores were 0.87 and 0.68 for the two tasks respectively which means that the submission was 5.75% worse than the winning entry for Subtask A and 5.88% worse than the winning entry for Subtask B.

As one of the intended use cases for the submitted systems was to serve as a baseline system in the future it is worth noting that they outperformed the provided baseline by the organizers (a linear SVN that scored 0.80 and 0.53 on the tasks) for both tasks.

Acknowledgments

We thank the Behr-Hella Thermocontrol GmbH for supporting this research. We also thank all reviewers and the competition organizers.

References

- Ricardo Cerri, Rodrigo C. Barros, and Andre CPLF de Carvalho. 2012. A genetic algorithm for hierarchical multi-label classification. In *Proceedings of the 27th annual ACM symposium on applied computing*, pages 250–255. ACM.
- Ricardo Cerri, Rodrigo C Barros, and André CPLF De Carvalho. 2014. Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences*, 80(1):39–56.
- Francesco Gargiulo, Stefano Silvestri, Mario Ciampi, and Giuseppe De Pietro. 2019. Deep neural network for hierarchical extreme multi-label text classification. *Applied Soft Computing*, 79:125–138.
- Eduardo Corrêa Gonçalves, Alex A Freitas, and Alexandre Plastino. 2018. A survey of genetic algorithms for multi-label classification. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.
- Matthew Honnibal and Ines Montani. 2019. spaCy library. <https://spacy.io>. Accessed: 2019-08-07.
- Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E. Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B. Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and et al. 2016. Jupyter notebooks - a publishing format for reproducible computational workflows. In *ELPUB*.
- Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM.

⁶<https://github.com/rother/germeval2019>

Wes McKinney. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, pages 51 – 56.

Travis E Oliphant. 2006. *A guide to NumPy*, volume 1. Trelgol Publishing USA.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Python Software Foundation. 2019. Python Programming Language. <https://python.org>. Accessed: 2019-08-07.

Steffen Remus, Rami Aly, and Chris Biermann. 2019. GermEval-2019 task 1: Shared task on hierarchical classification of blurbs. In *Proceedings of the GermEval 2019 Workshop*, Erlangen, Germany.

Seema Sharma and Deepti Mehrotra. 2018. Comparative analysis of multi-label classification algorithms. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 35–38. IEEE.

Kazuya Shimura, Jiyi Li, and Fumiyo Fukumoto. 2018. Hft-cnn: Learning hierarchical category structure for multi-label short text categorization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 811–816.

Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. 2008. Decision trees for hierarchical multi-label classification. *Machine learning*, 73(2):185.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*, pages 90–94. Association for Computational Linguistics.

Overview of GermEval Task 2, 2019 Shared Task on the Identification of Offensive Language

Julia Maria Struß

Potsdam University of
Applied Sciences
Kiepenheuerallee 5
14469 Potsdam

struss@fh-potsdam.de

Melanie Siegel

Darmstadt University of
Applied Sciences
Max-Planck-Str. 2
64807 Dieburg

melanie.siegel@h-da.de

Josef Ruppenhofer

Leibniz Institute for
German Language
R5, 6-13
68161 Mannheim

ruppenhofer@ids-mannheim.de

Michael Wiegand

Leibniz ScienceCampus
Heidelberg/Mannheim

wiegand@ids-mannheim.de

Manfred Klenner

University of Zurich
Andreasstrasse 15
8050 Zurich

klenner@cl.uzh.ch

Abstract

We present the second edition of the GermEval Shared Task on the Identification of Offensive Language. This shared task deals with the classification of German tweets from Twitter. Two subtasks were continued from the first edition, namely a coarse-grained binary classification task and a fine-grained multi-class classification task. As a novel subtask, we introduce the classification of offensive tweets as explicit or implicit.

The shared task had 13 participating groups submitting 28 runs for the coarse-grained task, another 28 runs for the fine-grained task, and 17 runs for the implicit-explicit task.

We evaluate the results of the systems submitted to the shared task. The shared task homepage can be found at <https://projects.fzai.h-da.de/iggsa/>

1 Introduction

The idea of social media was originally to enable an open exchange of information and opinions between people and thus to support communication. This idea of social participation is massively disturbed by current trends: Where an open exchange of views on political issues was possible, forums are increasingly inundated by offensive language. In many cases it is no longer possible to moderate forums without technical support.

The second GermEval Shared Task on the Identification of Offensive Language is intended to initiate and foster research on the identification of offensive content in German language microposts. Offensive comments are to be detected from a set of German tweets. We focus on Twitter, since tweets can be regarded as a prototypical type of micropost.

GermEval is a series of shared task evaluation campaigns that focus on natural language processing for the German language. Since 2014, there were shared tasks on named entity recognition, lexical substitution, sentiment analysis, hierarchical classification of blurbs, and identification of offensive language. These shared tasks have been run informally by self-organized groups of interested researchers and were endorsed by special interest groups within the German Society for Computational Linguistics (GSCL).

This paper will give a short overview on related work in §2. We will then describe the task in §3 and the data in §4. 13 teams participated in the shared task. We give an overview of their approaches and results in §5, and offer our conclusions in §6.

2 Related Work

For a recent overview of related work on the detection of abusive language, we refer the reader to Schmidt and Wiegand (2017) and Mishra et al. (2019). In what follows, we will briefly discuss related shared tasks as well as datasets for German.

- GermEval 2018 - To our knowledge this was the first shared task on the detection of offen-

sive language that included German language data. (Wiegand et al., 2018b)

- SemEval 2019 - Task 5 (HatEval) concerned multilingual (English and Spanish) detection of hate speech against immigrants and women in Twitter. The two subtasks addressed binary classification (hateful or not) and the classification of the target harassed as individual or generic. (Basile et al., 2019)
- SemEval 2019 - Task 6 (OffensEval 2019) was a shared task on identification and classification of offensive language in social media. The dataset contains 14,000 English tweets. The subtasks were to identify offensive tweets, to categorize them, and to identify the targets of the offensive posts. (Zampieri et al., 2019)
- Kaggle’s 2018 Toxic Comment Classification Challenge¹ was a shared task in which comments from the English Wikipedia are to be classified. There were 6 different categories of toxicity to be identified (i.e. *toxic*, *severe toxic*, *obscene*, *insult*, *identity hate* and *threat*). The categories were not mutually exclusive.
- The TRAC shared task on aggression identification (Kumar et al., 2018) included both English and Hindi Facebook comments. Participants had to detect abusive comments and to distinguish between *overtly aggressive comments* and *covertly aggressive comments*.
- The shared task on Automatic Misogyny Identification (AMI) (Fersini et al., 2018) is jointly run by IberEval² and EVALITA³. It exclusively focused on the detection of misogynist tweets on Twitter. There were two subtasks. Task A addressed the identification of misogynist tweets, while Task B focused on the categorization of misogynist tweets (i.e. *Discredit*, *Derailing*, *Dominance*, *Sexual Harassment & Threats of Violence*, *Stereotype & Objectification*, *Active* and *Passive*). Both IberEval and EVALITA included a task on English tweets. IberEval also included a task on Spanish tweets while EVALITA also featured a subtask on Italian tweets.

¹<https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge>

²<https://sites.google.com/view/iberEval-2018>

³<http://www.evalita.it/2018>

Most existing datasets of offensive language contain English data, such as the dataset described by Waseem and Hovy (2016). With regard to publicly-available German datasets for this task, we only know of Ross et al. (2016) who present a dataset of about 500 tweets which has been annotated regarding hate speech. The authors employed a binary categorization scheme. The dataset from Ross et al. (2016) may be too small for some data-hungry learning-based approaches. Being considerably larger, the German dataset produced for the GermEval shared tasks 2018 and 2019 with about 12,000 tweets in total should be a better alternative for such approaches.

3 Task Description

Participants were allowed to participate in one, two or all three subtasks and to submit at most three runs per task.

3.1 Subtask 1: Coarse-grained Binary Classification

Subtask 1 was to decide whether a tweet includes some form of offensive language or not. The tweets had to be classified into the two classes OFFENSE and OTHER. The OFFENSE category covered abusive language, insults, as well as merely profane statements.

3.2 Subtask 2: Fine-grained 4-way Classification

The second subtask involved four categories, a non-offensive OTHER class and three sub-categories of what is OFFENSE in subtask 1. In the case of PROFANITY, profane words are used, however, the tweet does not want to insult anyone. This typically concerns the usage of swearwords (*Scheiße*, *Fuck* etc.) and cursing (*Zur Hölle!* *Verdammt!* etc.). This can be often found in youth language. Swearwords and cursing may, but need not, co-occur with insults or abusive speech. Profane language may in fact be used in tweets with positive sentiment to express emphasis. Whenever profane words are not directed towards a specific person or group of persons and there are no separate cues of INSULT or ABUSE, then tweets are labeled as simple cases of PROFANITY.

In the case of INSULT, unlike PROFANITY, the tweet clearly wants to offend someone. INSULT is the ascription of negatively evaluated qualities or deficiencies or the labeling of persons as unworthy

(in some sense) or unvalued. Insults convey disrespect and contempt. Whether an utterance is an insult usually depends on the community in which it is made, on the social context (ongoing activity etc.) in which it is made, and on the linguistic means that are used (which have to be found to be conventional means whose assessment as insulting are intersubjectively reasonably stable).

And finally, in the case of ABUSE, the tweet does not just insult a person but represents the stronger form of abusive language. By abuse we define a special type of degradation. This type of degrading consists in ascribing a social identity to a person that is judged negatively by a (perceived) majority of society. The identity in question is seen as a shameful, unworthy, morally objectionable or marginal identity. In contrast to insults, instances of abusive language require that the target of judgment is seen as a representative of a group and it is ascribed negative qualities that are taken to be universal, omnipresent and unchangeable characteristics of the group. (This part of the definition largely co-incides with what is referred to as abusive speech in other research.) Aside from the cases where people are degraded based on their membership in some group, we also classify it as abusive language when dehumanization is employed even just towards an individual (i.e. describing a person as scum or vermin etc.).

3.3 Subtask 3: Implicit vs. Explicit Classification

Implicit offensive language is a form of offensive language where the expression of hate, condemnation, inferiority etc. as directed toward an explicitly or implicitly given target has to be inferred from the ascription of (hypothesised) target properties that are insulting, degrading, offending, humiliating etc. Rather than explicitly expressing their aversion, the writers hint at something degrading, i.e. their tweets imply that the target is unworthy etc.

Offensive tweets that use figurative language such as irony or sarcasm, or a play of words also count as implicit. Implicit offensive statements sometimes are only interpretable in their context. Also, inappropriate casual language while addressing a serious topic is subsumed under implicit offensive language.

The following examples⁴ illustrate our notion of

⁴These are examples from the GermEval 2018 corpus. We left misspellings untouched.

implicitness:

1. *Dem Kommentar entnehme ich das auch ihre Schaukel als Kind zu nahe an der Wand gestanden hat. (From the commentary I can see that your swing was too close to the wall as a child.)*
2. *Flüchtlinge fliehen nach Deutschland parallel dazu lassen sie ihre Familien in der Heimat sterben sehr ehrenhaft (Refugees flee to Germany at the same time they let their families die in their homeland very honourable ...)*
3. *Der arme ... Trauma jeden Tag , Sehnsucht nach Familiennachzug , kein eigenes Haus ... nachvollziehbar ... ! (The poor ... Trauma every day, longing for family reunion, no own house)*
4. *Es gibt nur ein Maas das ist ein Mittelmaß und heisst auch so @HeikoMaas (There is only one Maas that is a mediocrity and is also called so @HeikoMaas)*
5. *Also ich habe bei dem Herrn eine deutliche Alters Demenz gesehen. (Well I've seen that this man has an obvious age dementia)*

In example 1, it is the potential negative effect of a hypothesised situation that makes the reader understand the ascription of stupidity. Examples 2 and 3 are cases of sarcasm and irony, respectively. Neither is it honourable to leave someone in a dangerous situation (example 2) as the tweet states nor does a refugee suffer trauma just because they do not possess a house in their new host country (example 3). In example 4, a phonetic similarity between a name (*Maas*, the name of a German minister) and a negative concept (*Mittelmaß*, eng. *mediocrity*) suggests inferiority of the target (*Maas*). In example 5, the modal particle (*also*, eng. *well*) and a social distance indicating phrase (*dem Herrn*, eng. *this man*) are inappropriate in a discussion on such a topic (*Demenz*, eng. *dementia*). An honest diagnosis of a disease does not use such casual markers.

If the target is implicit, this might be an indicator of implicitness, but it is neither a necessary nor a sufficient condition. If a tweet comprises both implicit and explicit offensive language, we choose EXPLICIT as a label.

3.4 Evaluation Metrics

We evaluate the classification performance by the common evaluation measures *precision*, *recall*, and *F-score*. These measures are computed for each of the individual classes in the three subtasks. For each task, we also compute the *macro-average* precision, recall and F-score as well as the accuracy. We rank systems by their macro-average scores. We do not use accuracy for the ranking since in all three subtasks the class distribution is fairly imbalanced. Accuracy typically rewards correct classification of the majority class.

An evaluation tool computing all of the above mentioned evaluation measures on the three subtasks of the shared task was provided by the organizers prior to the release of the training data. It is publicly available and can be downloaded via the webpage of the shared task.

4 Data Set

As for last year’s task, Twitter was the source for our data collection. The reasons why we chose Twitter are a) that unlike other sources, Twitter contains a much higher proportion of offensive language (Wiegand et al., 2018a) and b) that, given the Twitter terms of service, we are able to make our collection publicly available.

4.1 Data Collection

The bulk of the available training data consisted of the training and test data from the first iteration of the shared task in 2018. We newly collected and annotated this year’s test data but also some further training data. To do so, we used the same approach to data collection that we had developed for the first iteration. That is, we sampled tweets from the timeline of various users rather than sampling randomly or on the basis of query term-based sampling. The latter two alternatives prove to be either too sparse in yielding offensive instances or too biased and lacking in variety.

We started by heuristically identifying users that regularly post offensive tweets. By sampling from the user’s timeline, we obtained offensive tweets that exhibited a more varied vocabulary than we would have obtained by sampling by pre-defined query terms. It also enabled us to extract a substantial amount of non-offensive tweets since only very few users post offensive content exclusively.

Since the majority of last year’s data came from the extreme right-wing spectrum and the dominant

topic concerned migration, we explicitly added timelines of users from the extreme left-wing spectrum to the 2019 data. Additionally we identified timelines discussing antisemitism in order to increase the topic variance in the data. Most of the user timelines considered for this topic can be assigned to the right-wing spectrum, but we also included timelines of users from other political directions, however we could not identify any timelines that can be assigned to the extreme left-wing spectrum concentrating on the topic. An overview of the data distribution with respect to the political orientation is given in Table 1.

Although this extraction process prevents the data set from becoming biased towards specific topics trending at the point in time when the extraction is run (a problem one typically faces when extracting data from the Twitter stream), we still found certain topics dominating our extracted data. However, this was not as extreme in the 2019 data as it was in the 2018 data, probably due to deliberately incorporating timelines of users from different political extremes. Most of the extracted offensive tweets in 2018 concerned the situation of migrants or the German government. The tweets not considered offensive, however, often addressed different topics. In the 2019 data we still found a stronger representation of certain political parties and some of their representatives, the government and the German state as well as some minorities in the offensive categories. For example, the politician names *Stegner* and *Maas* and the abbreviation *BRD* standing for ‘Federal Republic of Germany’ were much more often observed in offensive tweets. Since these high-frequency words undoubtedly do not represent offensive terms, we decided to *debias* our data collection by adding further tweets from the already collected timelines, belonging to the class OTHER and containing these terms. If this was not sufficient we added timelines from different political orientations to balance the topic over the classes (see Table 1). Because it was not always possible during the debiasing process to identify user timelines focusing on relevant topics from a different political spectrum, we also sampled further arbitrary tweets containing the terms in question. We specifically sought tweets from across the entire political spectrum. We also deliberately included tweets from users that regularly post highly-critical but inoffensive tweets with respect to the above topics. Otherwise, our data col-

topic/political orientation	ABUSE	INSULT	PROFANITY	OTHER	total
extreme left-wing	64	180	61	1802	2107
extreme right-wing	794	818	177	2137	3926
antisemitism	51	86	22	548	707
non-extreme (debiasing)	1		3	281	285
total	910	1084	263	4768	7025

Table 1: Distribution of topics/political orientation of the user timelines in the 2019 data

lection would allow classifiers to do well simply by recognizing offensive content as the combination of negative polarity and particular topics (e.g. *Stegner, Maas* or *BRD*).

When sampling tweets from Twitter, we also imposed certain formal restrictions on the tweets to be extracted (cf. Wiegand et al. (2018b)). They had to be a) written in German, b) contain at least five ordinary alphabetic tokens, c) contain no urls and d) be no retweets. These restrictions are mainly designed to speed up the annotation process (cf. §4.2) by removing tweets that are not relevant to the gold standard.

In splitting our data collection into training and test set, we made sure that any given user’s complete set of tweets was assigned to either the training set or the test set. This was done to avoid a situation where classifiers could benefit from learning user-specific writing styles or topic biases.

The data collection was also divided up in such a manner that the training and test sets have a similar class distribution. This is one of the major prerequisites for supervised learning approaches to work effectively.

The third subtask is based on the GermEval2018 data, namely those tweets from the 2018 shared task that are classified as abuse or insult (profanity was left out, because it is by definition explicit offensive language).

4.2 Annotation

4.2.1 Subtasks 1 and 2

Overall, we collected 7,025 new tweets for the 2019 Shared Task. Each of them was manually annotated by one of the organizers of the shared task. All annotators are native speakers of German.

In order to measure inter-annotation agreement, a sample of 300 tweets were annotated by all the annotators independently. We removed all tweets that were marked as HUNH or EXEMPT by at least one annotator. HUNH was used for incomprehensible utterances. We do not require that a

sentence is perfectly grammatically well-formed and correctly spelled to be included in our data. However, if a sentence is so erroneous that the annotator does not understand its content, then this sentence was labeled as HUNH and removed. This label also applies if the sentence is formally correct but the annotator still does not understand what is meant by this utterance. Tweets that were marked EXEMPT were ones that only contain abuse or insults representing the view of somebody other than the tweeter; utterances which depend on non-textual information; utterances that are just a series of hashtags and/or usernames, even if they indicate abusive speech (e.g. #crimigrants or #raprefugees); or utterances that are incomplete.

On the remaining 206 tweets, an agreement of $\kappa = 0.59$ was measured between the four annotators. It can be considered moderate (Landis and Koch, 1977). All remaining tweets of the gold standard were only annotated by one of the four annotators.

Table 2 displays the class distribution among the 2019 training and the test set. It comes as no surprise that non-offensive tweets represent the majority class. The most frequent subtype of offensive language are cases of (common) insult followed by abuse. By far the smallest category are profane tweets.

4.2.2 Subtask 3

After an initial phase, where we set up the guidelines, we chose 300 offensive tweets and four annotators classified each tweet as either implicit or explicit offensive language.

Our intention in this first round was to raise a common understanding of implicitness. After harmonization, 247 of the 300 tweets were classified as explicit offensive language (82.33%) while 52 (17.33%) were deemed to be implicit. See Table 3 for pairwise Kappa values.

As we expected, the annotation of implicitness is not an easy task. Accordingly, the agreement is

		training set		test set	
categories		freq	%	freq	%
coarse-grained	OFFENSE	1287	32.2	970	32.0
	OTHER	2707	67.8	2061	68.0
fine-grained	ABUSE	510	12.8	400	13.2
	INSULT	625	15.6	459	15.1
	PROFANITY	152	3.8	111	3.7
	OTHER	2707	67.8	2061	68.0
total		3994	100.0	3031	100.0

Table 2: Class distribution on the 2019 training and test set

	B	C	D
A	0.60	0.46	0.54
B		0.48	0.52
C			0.37

Table 3: Pairwise Kappa: 4 annotators, 300 tweets

moderate (most of the time). Two annotators, A and B, almost reached a substantial agreement, while annotators C and D only have a fair agreement. We thus decided to let A and B carry out a substantial part of the annotation.

The annotation of additional 1,800 examples resulted in a Kappa value of 0.51. After harmonization, the Kappa value of A and the gold standard was 0.60, while those of B and the gold standard was 0.82. The rest of the 2,890 tweets (600) were annotated by C and D. The agreement there was 0.42.

		freq	%
training set	IMPLICIT	259	13.20
	EXPLICIT	1699	76.80
test set	IMPLICIT	134	14.37
	EXPLICIT	798	75.63

Table 4: Class distribution subtask 3

Table 4 shows the class distribution for the training and the test data. The whole corpus comprises 8,541 tweets, 2,888 are offensive (33.81%) of which 393 (13.6%) were implicitly offensive.

4.3 Data Format

Our data is distributed in the form of tab-separated value files. An example row representing one tweet for subtasks 1 and 2 is shown in Table 5. As the task is focused only on the linguistic aspect of offensive language, each tweet is represented only by its text

in column 1. Meta-data contained in Twitter’s json files was not used. The text column is followed by the coarse-grained label in column 2 and the fine-grained label in column 3. Note that we applied no preprocessing to the tweet text with one exception: as shown in Table 5, line breaks were replaced with the special 5-character string |LBR| so that each tweet could be stored on one line.

For subtask 3 the data from 2018 was used. In order to provide for an additional layer distinguishing explicit from implicit offensive language, we added an additional column. Three labels are used: IMPLICIT, EXPLICIT or OTHER, see Table 6.

4.4 Sanity checks

To make sure empirically for subtasks 1 and 2 that the combination of last year’s data with this year’s data was sensible and there were no crucial differences that would actually harm performance, we performed an internal pre-test using last year’s winning system by TU Vienna (Padilla Montani and Schüller, 2018). We used all of last year’s data as well as this year’s new training data as the training set and tested on the new 2019 test set.

We performed a second sanity experiment after the task’s evaluation phase because it was only then noticed that there were erroneous labels on items of the 2019 training set. Altogether about 2.9% of the labels were affected: 15 cases of the class ABUSE, 28 cases of the class INSULT and 74 cases of the class OTHER. The PROFANITY class was not affected. We repeated the sanity check on a corrected version of the dataset to evaluate if the errors might have substantially harmed results in the competition.

The results for the initial sanity check on the original, slightly erroneous data are denoted by the rows coarse_e and fine_e in Tables 7 and 8, while the results for the run on the corrected data are denoted

@Ralf_Stegner Oman Ralle..dich mag ja immer noch keiner. Du willst das die Hetze gegen dich aufhört? LBR Geh in Rente und verzichte auf die 1/2deiner Pension	OFFENSE	INSULT
---	---------	--------

Table 5: Data format for subtask 1 and 2 (2019 dataset)

Der einzige, der sich noch darüber freut, dass Merkel auf ihrem Stuhl klebt LBR ist der beginnende Dekubitus.	OFFENSE	INSULT	IMPLICIT
--	---------	--------	----------

Table 6: Data format for subtask 3 (2018 dataset)

by the rows rows coarse_c and fine_c. Overall, the results for these sanity checks are very similar to the system’s results on last year’s tasks, regardless of whether we the training set includes a slight number of errors or not. (The corrected version of the training set is also now publicly available from the shared task homepage.)

For subtask 3, no sanity checks were needed.

5 Submissions and Results

The full set of results for all three subtasks is available at the shared task website.

Table 9 presents descriptive statistics for the scores produced in this year’s and last year’s iterations of subtask 1 and 2. For subtask 1, coarse-grained classification, we can see that this year the participants’ scores are more tightly clustered, yielding a lower standard deviation. For subtask 2, the fine-grained task, there was no similar development.

5.1 Coarse-grained Classification

We received 28 different runs from 12 teams for the binary classification into OFFENSE vs. OTHER. For lack of space, we only show the best 15 runs in Table 10. Compared to the previous year, this year’s winning F-score is higher, but very slightly so (76.95 vs. 76.77). Of course, these number cannot be compared directly as they involved different training and test sets.

5.2 Fine-grained Classification

For the second subtask we received 28 different runs from 12 teams for the fine-grained classification. For lack of space, we only show the best 10 runs in Table 11. Compared to last year’s results, the winning score is higher by about 0.9% F-score. As in the case of the coarse grained subtask, this cannot be readily interpreted without further investigation.

5.3 Implicit vs. Explicit Classification

Seven groups participated in subtask 3, which was a difficult subtask. Although the best accuracy was 86.77% with a F-score of 73.11, the numbers for the class IMPLICIT were low, the best F-score being 53.93. The subtask is difficult due to the skewed distribution of that class, just 13.9% of the offensive tweets are labeled as implicit.

5.4 General Conclusions Drawn from the Evaluation

5.4.1 System Design

Although in terms of absolute F-scores, the best performing system on all 3 subtasks was a system that employed some form of the latest transformer based language model BERT (Devlin et al., 2019) (i.e. *UPB* on subtasks 1 and 2; *hpiDEDIS* on subtask 3), at least on subtasks 1 and 2 there were systems which did not incorporate BERT (i.e. *TUWienKBS* on subtask 1 and *FoSIL* on subtask 2) but still performed very well (that is within 1% point of the top performing system). Only on subtask 3 is there a larger difference between the best performing system and the best system not employing BERT, i.e. *FoSIL*, with a gap of more than 3.5% points.

BERT seems to be generally effective. All 3 teams that participated in the shared task and incorporated some form of BERT (*UPB*, *hpiDEDIS* and *bertZH*) were among the top performing systems. The variation of BERT that consistently outperformed the other ones is a model pre-trained on 6 million German tweets (*UPB*). The other two teams just fine-tuned the existing pre-trained models.

Surprisingly, for all 3 subtasks there is no system among the top 3 teams which employs *standard* deep-learning architectures, such as LSTMs or CNN. Instead, with *FoSIL* we still find systems that are based on traditional classifiers, such as SVMs. This year’s results are also mostly consis-

	OFFENSE			OTHER			average		
	P	R	F	P	R	F	P	R	F
coarse _e	67.23	69.18	68.19	85.29	84.13	84.71	76.26	76.65	76.46
coarse _c	68.20	68.76	68.48	85.24	84.91	85.08	76.72	76.84	76.78

Table 7: Results of sanity checks on error-containing and clean test data in coarse setting

		P	R	F
fine _e	ABUSE	47.29	41.50	44.21
	INSULT	44.71	36.82	40.38
	OTHER	83.33	88.26	85.72
	PROFANITY	42.02	45.05	43.48
	average	54.34	52.91	53.61
fine _c	ABUSE	47.41	41.25	44.12
	INSULT	45.76	38.78	41.98
	OTHER	83.65	88.40	85.96
	PROFANITY	43.10	45.05	44.05
	average	54.98	53.37	54.16

Table 8: Results of sanity checks on error-containing and clean test data in fine-grained setting

tent with last year’s results: the best performing systems incorporated some form of word embeddings and some information on the subword level (e.g. character n-grams). Ensemble methods may be effective (*TUWienKBS*) but they seem not to be a crucial ingredient for high scores. The same holds for task-specific lexicons. Of the 3 top-performing systems on the 3 subtasks, only *FoSIL* employed that type of information.

In subtask 3, the best performing systems (*hpi-DEDIS*, *UPB*, rank 1-5 with various runs) were using BERT as a resource (fine-tuned it). Of the 7 participants, 5 used neural approaches (including BERT), i.e. RNN (*inriaFBK*), CNN (*fkie*, *HAU*) and LSTM (*fkie*). Two worked with German FastText (*RGCL*, *FoSIL*), one also considered a Random Forest approach and one also submitted a SVM based run.

5.4.2 Task and Data

With regard to subtask 1, if we compare the difference between the F-score of the best performing system to the median between this year (median: 72.95; best system: 76.95) and last year (median: 69.15; best system: 76.77), we find that the median has risen appreciably (by more than 3% points) while the best score has maintained its level of performance. From that we may conclude that the

average system that took part in this year’s edition of the shared task is notably stronger than last year.

In terms of the best overall scores that have been achieved in subtasks 1 and 2 in this year’s edition of the shared task, there is hardly any improvement. We re-trained last year’s winning system on this year’s training data and compared the classification on this year’s test data (cf. Tables 7 and 8) with the best performing system in this year’s competition (cf. Tables 10 and 11). Surprisingly, we obtained only marginally worse results with last year’s system (subtask 1: 76.46 vs. 76.95; subtask 2: 53.61 vs. 53.95). Given that this year’s training set was larger, this could mean one of two things. First, the additional data might not have helped even though test and training data were otherwise similar because the system was not able to make use of relevant features. Alternatively, the increase in data this year might have been offset by the new data being more difficult so that overall the system reached only the same level of performance as last year. These questions can best be addressed by running the same system on various combinations of this and last year’s data, which unfortunately is outside the scope of this overview paper.

All in all, these results underline that the problem of offensive language detection is far from solved. It also suggests that a thorough error analysis is required. Only thus can we learn which systematic errors even the best performing systems make and, hopefully, get ideas how to devise new methods which even solve these types of phenomena.

6 Conclusion

In this paper, we described the second edition of the GermEval Shared on the Identification of Offensive Language. The shared task comprised three tasks, a coarse-grained binary classification task, a fine-grained multi-class classification task and a novel classification task in which explicit tweets had to be separated from implicit ones. In total, 13 groups participated in the shared task submitting 28 runs for each of the first two subtasks and 17 runs for the last subtask.

year	subtask	# teams	# runs	min	max	median	mean	sd
2019	coarse	12	28	54.87	76.95	72.95	71.51	5.67
	fine	12	28	36.83	53.59	46.55	46.63	5.18
	implicit/explicit	7	17	55.37	73.11	68.87	67.19	5.26
2018	coarse	20	51	49.03	76.77	69.15	66.35	8.45
	fine	11	25	32.07	52.71	38.75	39.71	5.00

Table 9: Summary statistics for overall macro F1-scores in the three subtasks and as a reference the figures of last year’s edition

While for the third subtask, the data of the previous edition were augmented by the classification scheme of this new task, for the first and second subtask, completely new tweets were added to the collection. For these two subtasks, we added about 4,000 manually labeled training tweets. Similar to last year, much care was taken in order to provide a relatively unbiased dataset. Unlike the data from the previous edition, the new data also contain offensive language originating from other areas than the extreme right.

Approaches that were effective in last year’s edition, such as supervised classifiers using word embeddings, subword information and ensemble methods, also proved effective in this year’s edition. However, similar effectiveness without less task-specific design could be achieved by classifiers based on the recent BERT model.

Surprisingly, the best system of this year’s system on the coarse-grained task is on a par of last year’s winning system. This result again underlines the difficulty of this task. Further error analyses should be carried in order to determine which types of errors even the best performing systems incur. This would hopefully provide fruitful research directions for future work.

Acknowledgments

We are grateful to the large number of participants whose enthusiastic participation made GermEval 2019 a success. We would like to thank Joane Amrhein for maintaining the shared task home page and mailing lists and supporting the evaluation process. We would also like to thank Sophia Conrad (BA student), Andreas Säuberli (BA student) and Noëmi Aepli (PhD student) for their contribution to subtask 3. All annotators are native speakers of German. We also thank the Konvens 2019 conference organizers for their support. Michael Wiegand was supported by the Leibniz ScienceCampus “Empirical Linguistics and Computational Modeling”,

funded by the Leibniz Association under grant no. SAS-2015-IDS-LWC and by the Ministry of Science, Research, and Art (MWK) of the state of Baden-Württemberg.

References

- Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.
- Elisabetta Fersini, Paolo Rosso, and Maria Anzovino. 2018. Overview of the task on automatic misogyny identification at ibereval 2018. In *IberEval@SEPLN*, pages 214–228.
- Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.
- J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174.
- Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. 2019. Tackling Online Abuse: A Survey of Automated Abuse Detection Methods. *arXiv e-prints*, August.
- Joaquin Padilla Montani and Peter Schüller. 2018. TUWienKBS at GermEval 2018: German Abusive Tweet Detection. *Proceedings of GermEval 2018*,

	team	submission	runID	Accuracy			OFFENSE			OTHER			average
				percent	correct	total	P	R	F	P	R	F	
1	UPB	coarse_1	79.38	2406	3031	66.26	72.47	69.23	86.45	82.63	84.50	76.35	77.55
2	UPB	coarse_3	79.38	2406	3031	66.26	72.47	69.23	86.45	82.63	84.50	76.35	77.55
3	UPB	coarse_2	79.64	2414	3031	67.53	70.10	68.79	85.67	84.13	84.90	76.60	77.12
4	TUWienKBS	coarse_1	80.04	2426	3031	69.73	66.49	68.07	84.57	86.41	85.48	77.15	76.45
5	TUWienKBS	coarse_2	79.94	2423	3031	69.34	66.91	68.10	84.68	86.07	85.37	77.01	76.49
6	FoSIL	coarse_2	79.54	2411	3031	67.68	69.07	68.37	85.30	84.47	84.89	76.49	76.63
7	FoSIL	coarse_1	79.25	2402	3031	66.73	70.10	68.38	85.59	83.55	84.56	76.16	76.83
8	hpIDEDIS	coarse_2	78.69	2385	3031	64.86	72.89	68.64	86.45	81.42	83.86	75.66	77.15
9	hpIDEDIS	coarse_3	79.71	2416	3031	70.66	62.58	66.38	83.29	87.77	85.47	76.98	75.18
10	hpIDEDIS	coarse_1	76.97	2333	3031	61.54	74.74	67.50	86.78	78.02	82.17	74.16	76.38
11	inriaFBK	coarse_2	78.55	2381	3031	67.74	62.99	65.28	83.14	85.88	84.49	75.44	74.44
12	hsrl	coarse_1	78.39	2376	3031	68.00	61.34	64.50	82.61	86.41	84.47	75.30	73.88
13	inriaFBK	coarse_1	78.55	2381	3031	69.18	59.48	63.97	82.11	87.53	84.73	75.65	73.51
14	rgcl	coarse_2	77.96	2363	3031	81.72	40.10	53.80	77.26	95.78	85.53	79.49	67.94
15	rgcl	coarse_3	77.37	2345	3031	83.49	36.49	50.79	76.37	96.60	85.30	79.93	66.55

Table 10: Top 15 runs for subtask 1: coarse-grained classification

	team	submission	runID	accuracy			ABUSE			INSULT			OTHER			PROFLTY			average	
				percent	correct	total	P	R	F	P	R	F	P	R	F	P	R	F		
1	UPB	fine_1	73.61	2231	3031	44.04	60.00	50.79	49.49	32.03	38.89	84.69	88.55	86.57	55.88	17.12	26.21	58.53	49.42	
2	FoSIL	fine_2	71.36	2163	3031	42.57	58.75	49.37	45.30	45.10	45.20	85.88	82.63	84.22	46.15	16.22	24.00	54.98	50.67	
3	FoSIL	fine_1	72.02	2183	3031	42.60	58.25	49.21	46.81	38.34	42.16	84.60	85.30	84.95	53.33	14.41	22.70	56.83	49.08	
4	berZH	fine_2	74.46	2257	3031	55.29	45.75	50.07	48.09	41.18	44.37	83.43	90.15	86.66	33.75	24.32	28.27	55.14	52.67	
5	UPB	fine_2	71.66	2172	3031	43.33	58.50	49.79	43.51	39.43	41.37	85.67	84.13	84.90	45.10	20.72	28.40	54.40	50.70	
6	UPB	fine_3	73.57	2230	3031	45.61	54.50	49.66	47.21	36.82	41.37	84.69	88.55	86.57	45.00	16.22	23.84	55.63	49.02	
7	TUWienKBS	fine_2	71.86	2178	3031	45.36	41.50	43.34	44.78	38.34	41.31	82.62	87.19	84.84	40.21	35.14	37.50	53.24	50.54	
8	berZH	fine_1	73.38	2224	3031	54.55	43.50	48.40	43.42	40.96	42.15	82.63	89.33	85.85	41.18	18.92	25.93	55.44	48.18	
9	TUWienKBS	fine_1	73.18	2218	3031	46.20	38.00	41.70	49.82	30.94	38.17	80.58	91.80	85.82	46.38	28.83	35.56	55.75	47.39	
10	hpIDEDIS	fine_3	70.04	2123	3031	44.94	47.75	46.30	39.19	48.58	43.39	85.19	81.76	83.44	40.68	21.62	28.24	52.50	49.93	51.18

Table 11: Top 10 results for subtask 2: fine-grained classification

14th Conference on Natural Language Processing (KONVENS 2018), Vienna, Austria September 21, 2018.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Watzki. 2016. Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis. In *Proceedings of the Workshop on Natural Language Processing for Computer-Mediated Communication*, pages 6–9, Bochum, Germany.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the EACL-Workshop on Natural Language Processing for Social Media (SocialNLP)*, pages 1–10, Valencia, Spain.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Michael Wiegand, Josef Ruppenhofer, Anna Schmidt, and Clayton Greenberg. 2018a. Inducing a Lexicon of Abusive Words – A Feature-Based Approach. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*, New Orleans, USA.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018b. Overview of the GermEval 2018 shared task on the identification of offensive language. Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018), Vienna, Austria September 21, 2018, pages 1 – 10. Austrian Academy of Sciences, Vienna, Austria.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offensEval). In *SemEval@NAACL-HLT*.

7 Appendix

group id	authors	affiliation	paper title
InriaFBK	Michele Corazza, Stefano Menini, Elena Cabrio, Sara Tonelli and Serena Villata	Inria, France and Fondazione Bruno Kessler, Italy	InriaFBK Drawing Attention to Offensive Language at GermEval2019
hshl	Kristian Rother and Achim Rettberg	Hochschule Hamm-Lippstadt, Germany	German Hatespeech classification with Naive Bayes and Logistic Regression - hshl at GermEval 2019 - Task 2
fkie	Theresa Krumbiegel	Fraunhofer FKIE, Germany	FKIE - Offensive Language Detection on Twitter at GermEval 2019
FraunhoferSIT	Inna Vogel and Roey Regev	Fraunhofer Institute SIT, Germany	FraunhoferSIT at GermEval 2019: Can Machines Distinguish Between Offensive Language and Hate Speech? Towards a Fine-Grained Classification
bertZH	Tim Graf and Luca Salini	University of Zurich, Switzerland	bertZH at GermEval 2019: Fine-Grained Classification of German Offensive Language using Fine-Tuned BERT
FoSIL	Florian Schmid, Justine Thielemann, Anna Mantwill, Jian Xi, Dirk Labudde and Michael Spranger	University of Applied Sciences Mittweida, Germany	FoSIL - Offensive language classification of German tweets combining SVMs and deep learning techniques
h.da	Isabell Börner, Midhad Blazevic, Maximilian Komander and Margot Mieskes	Darmstadt University of Applied Sciences, Germany	2019 GermEval Shared Task on Offensive Tweet Detection h_da submission
HAU	Johannes Schäfer, Tom De Smedt and Sylvia Jaki	University of Hildesheim, Germany and University of Antwerp, Netherlands	HAU at the GermEval 2019 Shared Task on the Identification of Offensive Language in Microposts: System Description of Word List, Statistical and Hybrid Approaches
UPB	Andrei Paraschiv and Dumitru-Clementin Cercel	University Politehnica of Bucharest, Romania	UPB at GermEval-2019 Task 2: BERT-Based Offensive Language Classification of German Tweets
hpiDEDIS	Julian Risch, Anke Stoll, Marc Ziegelle and Ralf Krestel	Hasso Plattner Institute, Heinrich Heine University Düsseldorf and University of Passau, Germany	hpiDEDIS at GermEval 2019: Offensive Language Identification using a German BERT model
HUIU	Melanie Andresen, Melitta Gillmann, Jowita Grala, Sarah Jablotschkin, Lea Röseler, Eleonore Schmitt, Lena Schnee, Katharina Straka, Michael Vauth, Sandra Kübler and Heike Zinsmeister	Universität Hamburg (Germany), Universitt Bamberg (Germany), Indiana University (USA)	The HUIU Contribution for the GermEval Shared Task 2
TUWienKBS19	Joaqun Padilla Montani and Peter Schüller	TU Wien, Austria	TUWienKBS19 at GermEval Task 2, 2019: Ensemble Learning for German Offensive Language Detection
RGCL	Alistair Plum, Tharindu Ranasinghe, Constantin Orasan and Ruslan Mitkov	University of Wolverhampton, UK	RGCL at GermEval 2019: Offensive Language Detection with Deep Learning

Table 12: Group IDs, Authors and Paper Titles

InriaFBK Drawing Attention to Offensive Language at Germeval2019

Michele Corazza[†], Stefano Menini[‡]

Elena Cabrio[†], Sara Tonelli[‡], Serena Villata[†]

[†]Université Côte d’Azur, CNRS, Inria, I3S, France

[‡]Fondazione Bruno Kessler, Trento, Italy

michele.corazza@inria.fr

{menini, satonelli}@fbk.eu

{elena.cabrio, serena.villata}@unice.fr

Abstract

In this paper we describe the system developed by InriaFBK team and submitted to the Germeval2019 task on offensive language detection and classification. With the same architecture we participate to all subtasks: binary classification of offensive and not offensive tweets, 4-class message categorisation based on offense type (Profanity, Insult, Abuse and Other), and classification of explicit and implicit offensive language. The two runs submitted for each subtask are obtained with and without attention mechanism. After evaluating our system performance on Germeval2018 test set, we observe that attention is remarkably beneficial in the more challenging tasks of implicit offense detection and offense categorisation.

1 Introduction

Detecting hurtful, derogatory and obscene comments online has become of paramount importance for the well-being of users, who access social networks to exchange ideas and build a sense of community, as well as for social media platforms, which have been accused of fostering the widespread of hurtful content. Recent initiatives at institutional level have been undertaken to limit the phenomenon of online hate speech, see for example the 2018 Code of Conduct signed between EU representatives and four major social media players¹. The monitoring process following the adoption of this code of conduct has shown that, when users report offensive content, 88.9% of them get a reply from the social media platform within 24 hours. However, if we compare Facebook, YouTube, Instagram and Twitter, statistics show that the latter

tends to remove remarkable less content than the others, i.e. only 43.5% of reported messages compared to 71.7% by the other three platforms on average.² The EU report highlights how most of the feedback from Twitter is on *trusted reports* rather than on general users reports, making reasonable to think that Twitter policies are less restrictive and do not aim to comply with every user. This makes the issue of automatic hate speech detection on Twitter even more urgent, especially when developed systems are able to identify different types of offense and cope with implicit hate messages.

In this paper, we present our system submitted to the Germeval 2019 task for offensive language detection, and detail the two runs for each of the three subtasks (with and without attention mechanism). The general framework is an improved version of the InriaFBK system developed for Italian hate speech detection (Corazza et al., 2018a) and for German at Germeval 2018 (Corazza et al., 2018b), with a more careful choice of external embeddings and of neural network parameters. Furthermore, we evaluate the contribution of attention mechanism to each of the subtasks.

2 Related work

Many solutions and resources are available to perform hate speech detection and classification on English data. For example, in Waseem and Hovy (2016) the authors not only present their work on the classification of racist and sexist tweets adopting a logistic regression model based on one-to-four-character n-grams, but also release an annotated dataset for the task. Other classification approaches have been tested on the same dataset, see for example (Kshirsagar et al., 2018) proposing a neural classifier using pre-trained word embeddings and max/mean pooling from fully-

¹<https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32008F0913&format=EN>

²https://ec.europa.eu/commission/news/counteracting-illegal-hate-speech-online-2019-feb-04_en

connected transformations of these embeddings. The approach is tested also on the the HATE dataset (Davidson et al., 2017) and on the Harassing dataset (Golbeck et al., 2017) showing that it is able to learn associations among words typically used in hateful communication.

Other approaches targeting languages different from English have been proposed mainly in the context of shared tasks, such as the first *Hate Speech Detection* (HaSpeeDe) task for Italian (Bosco et al., 2018) and *Aggressiveness Detection* (Carmona et al., 2018) for Mexican Spanish at IberEval 2018.

As for German, the first shared task was proposed in 2018 on the *Identification of Offensive Language* (Wiegand et al., 2018). The task covers the detection of offensive comments from a set of German tweets, that had to be further classified into abusive language, insults and profane statements. The systems presented by the participants introduce a number of different approaches, ranging from feature-based supervised learning (i.e., SVMs for the top-performing system TUWienKBS (Padilla Montani and Schüller, 2018)) to deep learning. Most top performing systems in both subtasks are based on deep learning, such as spMMMP (von Grunigen et al., 2018), uhhLT (Wiedeman et al., 2018), SaarOffDe (Stammbach et al., 2018), InriaFBK (Corazza et al., 2018b).

Looking at the systems participating in the above tasks, we observe a number of features shared by many deep learning approaches, such as domain-specific word embeddings, the use of emotion or sentiment lexica, features related to the message (e.g. length, punctuation marks, etc.) as well as specific pre-processing steps.

3 Data and Tasks

At the Germeval evaluation, three different subtasks were proposed: one for the detection of offensive messages, one for a fine-grained classification in four classes, namely *Profanity*, *Insult*, *Abuse* and *Other*, and one for the identification of explicit and implicit hate. Since subtask I and II had already been proposed at Germeval2018, the organisers allowed participants to use as training data the data released both in 2018 and 2019 as training data. For the third subtask, instead, the dataset was novel.

For the submissions of subtask I and II, we use the concatenated Germeval 2018 and 2019 training sets for training and the Germeval2018 test data as

validation set. For subtask III we isolate 20% of the training set for validation (see details in Section 4). Below we summarise the number of instances used as training for each subtask:

Subtask I - Binary classification: The two labels are ‘offensive’ and ‘other’. The latter was reserved for tweets which were not offensive. The binary classification subtask involved 2,975 messages with ‘offensive’ label and 6,029 messages with the ‘other’ label.

Subtask II - Fine-grained classification: The four classes annotated are ‘profanity’, ‘insult’, ‘abuse’ and ‘other’. In the corpus, there are 1,220 messages for ‘insult’, 223 for ‘profanity’, 1,532 for ‘abuse’, and 6,029 messages for ‘other’.

Subtask III - Implicit and Explicit offense classification: All messages in this dataset are offensive, but they are labeled either as ‘implicit’ or ‘explicit’. In particular, there are 1,699 explicitly offensive message, and 259 implicit ones.

To compute the preliminary evaluation results reported in this paper, instead, we change the splits by using 20% of the 2018 and 2019 training sets for subtasks I and II for validation, and compute the performance reported in the following tables on the Germeval 2018 test set. For subtask III, we use 20% of the training set for validation and 20% as test set.

4 System Description

In order to perform an analysis of the activations of an attention mechanism, we use a recurrent neural network with attention applied to the outputs of the recurrent GRU layer, and compare its performance to the same network with no attention applied. Since the domain of the task is interactions on a social media platform, we apply some ad-hoc preprocessing steps, which are detailed in the next subsection, in order to improve the performance of the classifier on Twitter-specific language.

To isolate the validation set from the training data, we use `train_test_split` from scikit-learn (Pedregosa et al., 2011).

4.1 Preprocessing

Since the language of social media interactions presents unique challenges for standard NLP tasks, we normalise the tweet content by replacing user mentions and URLs with the strings “username”

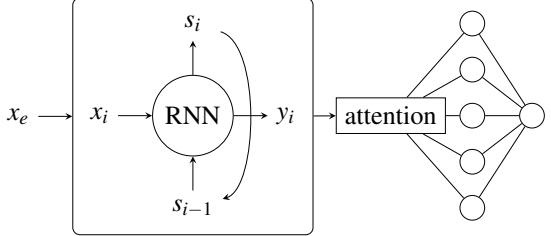


Figure 1: The recurrent neural architecture

and ‘‘URL’’ respectively. We do not apply hashtag splitting, since it proved not effective on German in a comparative evaluation for hate speech detection (Corazza et al., 2019).

4.2 Word embeddings

Word embeddings (Mikolov et al., 2013; Pennington et al., 2014) are a widely used approach to represent word meaning in natural language processing tasks, as they allow to acquire some information about words through an unsupervised process. However, word embedding resources have a major drawback when it comes to processing German data, since they may not contain all compounds or all the declensions of a single word, resulting in many out-of-vocabulary terms. This issue can be alleviated by using subword information to represent a term as the sum of the vectors representing its character n-grams. This is the main reason why we chose to use FastText embeddings (Bojanowski et al., 2016), pretrained on Common Crawl and Wikipedia³.

4.3 Recurrent model

We develop a simple recurrent neural network model and use it for all subtasks. We use the word embeddings from the words of each tweet as input for a GRU (Cho et al., 2014) of size 100. Recurrent dropout of 0.2 is applied to the GRU. The output at the last timestep from the GRU is then fed to a single, fully-connected layer with 200 neurons, followed by one or more output neurons, depending on the subtask. For subtasks 1 and 3 a single, sigmoid activated neuron is used, while for subtask 2 we use four outputs with a softmax activation. The binary subtasks use binary crossentropy as the loss function, while subtask 2 uses categorical crossentropy. The optimizer used is Adam and the models were implemented in Keras (Chollet and others, 2015). In addition to classifying offensive

Category	Precision	Recall	F1 Score
No Attention			
Offensive	0.677	0.630	0.653
Other	0.831	0.859	0.845
Macro AVG	0.754	0.744	0.749
+ Attention			
Offensive	0.692	0.595	0.640
Other	0.821	0.875	0.847
Macro AVG	0.757	0.735	0.746

Table 1: Subtask 1 without attention (above) and with attention (below)

language, our goal was also to examine how an attention mechanism could improve performance, and whether the activations could be used to understand the classifier behavior. In particular, we consider the output of the GRU layer g :

$$GRU(x) = (e_1, e_2, \dots, e_n) \quad e_i \in \mathbb{R}^{100} \quad (1)$$

We then apply a perceptron layer to each of the outputs of the GRU and use softmax to obtain weights that sum to 1:

$$A(e) = softmax(F(e)) \quad (2)$$

Where:

$$\begin{aligned} F(e) &= (f(e_1), \dots, f(e_n)) \\ f(e_i) &= (We_i + b) \\ W &\in \mathbb{R}^{1 \times 100} \quad b \in \mathbb{R}^1 \end{aligned} \quad (3)$$

After applying a perceptron layer to each output of the GRU, we use a softmax layer so that the sum of all timesteps is one (padding is ignored). The weights obtained are then multiplied elementwise with the outputs of the GRU:

$$a(x) = A(GRU(x)) \odot GRU(x) \quad (4)$$

We then sum over the vectors obtained by applying attention to the outputs of the GRU, and use the resulting vector to classify offensive language, by feeding it to a single, fully connected hidden layer followed by the outputs.

5 Evaluation

For subtasks I and II, we report below the results obtained on the Germeval 2019 test set, comparing the system performance with and without attention mechanism.

With respect to subtask I (see Table 1), the two models perform similarly well. In particular, while the model without attention is the better performing one with respect to the offensive class, the F1

³[https://fasttext.cc/docs/en/crawl-veectors.html](https://fasttext.cc/docs/en/crawl-vectors.html)

Category	Precision	Recall	F1 Score
No Attention			
Abuse	0.371	0.455	0.409
Insult	0.375	0.397	0.385
Profanity	0.355	0.099	0.155
Other	0.832	0.817	0.825
Macro AVG	0.483	0.442	0.462
+ Attention			
Abuse	0.443	0.503	0.470
Insult	0.425	0.325	0.367
Profanity	0.475	0.171	0.252
Other	0.824	0.874	0.849
Macro AVG	0.542	0.468	0.502

Table 2: Subtask II without attention(above) and with attention mechanism (below)

metric on the “other” class is remarkably similar, with a slight advantage for the model with attention. This results in a slight advantage in terms of macro average F1 for the model without attention.

For Subtask II (see Table 2), focusing on fine-grained classification, the observed behaviour of the two models is still similar, but this time the attention-based model outperforms the attentionless one across all categories except for the “insult” one, showing that attending over single words can be useful when classifying different types of offensive language. The largest improvement is achieved on the Profanity class, showing that attention mechanism in this case can better learn from few examples (only 223 for this class), while it is less evident on the Other class, which is the majority one (6,029 training instances).

Category	Precision	Recall	F1 Score
No Attention			
Explicit	0.891	0.964	0.926
Implicit	0.580	0.299	0.394
Macro AVG	0.735	0.631	0.679
+ Attention			
Explicit	0.910	0.918	0.914
Implicit	0.488	0.463	0.475
Macro AVG	0.699	0.690	0.695

Table 3: Subtask III without attention (above) and with attention mechanism (below)

With respect to subtask III (see Table 3), we observe a more significant difference between the two models on the Implicit class, while the Explicit one is equivalent. This may confirm the model behavior observed in subtask II, where classes with less examples had improved performance when using attention. Also in this case, the model using attention has a higher F1 score value for the implicit class, for which only 259 training instances are available.

6 Attention activations

In order to understand how attention affects the classification outcome and whether the outputs of the attention layer can help explain the classification performed by our model, we examined the attention for each word in the test set of Germeval 2018 (the outputs of Equation 2), using a model trained on the first subtask. Looking at the lemmas ranked by average weights learned by the attention mechanism, we observe that the top ones are mostly emotionally loaded with a negative connotation. For example, we find among the lemmas with highest weights words such as *klatsch*, *Opportunistin*, *Elektrojude*, and *verpisst*. Looking at attention weights of the words composing a tweet, we observe the same trend: in the messages correctly classified as ‘Offensive’ the words with highest attention weights are those with negative polarity, that mostly contribute to correct classification. For example, in *Von mir aus könnt ihr jämmerlich verrecken* the last two words have the highest attention. In a similar way, the last word of the following tweet is the one with highest attention weight: *Ich persönlich scheisse auf die grüne Kinderfickerpartei*. These findings suggest that the attention mechanism is effectively capturing the words whose meaning and polarity most contribute to the classifier choice. Furthermore, examining activation weights can lead to precious insight into the inner criteria used by models to detect offensive language.

7 Conclusions

In this work we detailed the system runs submitted by the InriaFBK team to Germeval 2019. With the same architecture we participated in all three subtasks, performing both binary and multi-class classification. In a comparative evaluation, our results show that the attention mechanism has a positive impact on classes with few training instances, while it has no remarkable effect on classes that are well represented in the training set.

Acknowledgments

Part of this work was funded by the CREEP project (<http://creep-project.eu/>), a Digital Wellbeing Activity supported by EIT Digital in 2018. This research was also supported by the HATEMETER project (<http://hatemeter.eu/>) within the EU Rights, Equality and Citizenship Programme 2014-2020.

References

- [Bojanowski et al.2016] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*.
- [Bosco et al.2018] Cristina Bosco, Felice Dell’Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA 2018 hate speech detection task. In *Proceedings of the Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018) co-located with the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), Turin, Italy.*
- [Carmona et al.2018] Miguel Ángel Álvarez Carmona, Estefanía Guzmán-Falcón, Manuel Montes-y-Gómez, Hugo Jair Escalante, Luis Villaseñor Pineda, Verónica Reyes-Meza, and Antonio Rico Sulayes. 2018. Overview of MEX-A3T at ibereval 2018: Authorship and aggressiveness analysis in mexican spanish tweets. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018) co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018), Sevilla, Spain, September 18th, 2018.*, pages 74–96.
- [Cho et al.2014] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar; A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.
- [Chollet and others2015] François Chollet et al. 2015. Keras. <https://keras.io>.
- [Corazza et al.2018a] Michele Corazza, Stefano Menini, Pinar Arslan, Rachele Sprugnoli, Elena Cabrio, Sara Tonelli, and Serena Villata. 2018a. Comparing different supervised approaches to hate speech detection. In *Proceedings of the Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018) co-located with the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), Turin, Italy.*
- [Corazza et al.2018b] Michele Corazza, Stefano Menini, Arslan Pinar, Rachele Sprugnoli, Cabrio Elena, Sara Tonelli, and Villata Serena. 2018b. Inri-afb at germeval 2018: Identifying offensive tweets using recurrent neural networks. In *Proceedings of GermEval 2018*, pages 80–84.
- [Corazza et al.2019] Michele Corazza, Stefano Menini, Elena Cabrio, Sara Tonelli, and Serena Villata. 2019. Robust hate speech detection: A cross-language evaluation. *Under review.*
- [Davidson et al.2017] Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017.*, pages 512–515.
- [Golbeck et al.2017] Jennifer Golbeck, Zahra Ashktorab, Rashad O. Banjo, Alexandra Berlinger, Siddharth Bhagwan, Cody Buntain, Paul Cheakalos, Alicia A. Geller, Quint Gergory, Rajesh Kumar Gnanasekaran, Raja Rajan Gunasekaran, Kelly M. Hoffman, Jenny Hottle, Vichita Jienjtitlert, Shivika Khare, Ryan Lau, Marianna J. Martindale, Shalmali Naik, Heather L. Nixon, Piyush Ramachandran, Kristine M. Rogers, Lisa Rogers, Meghna Sardana Sarin, Gaurav Shahane, Jayanee Thanki, Priyanka Vengataraman, Zijian Wan, and Derek Michael Wu. 2017. A large labeled corpus for online harassment research. In *Proceedings of the 2017 ACM on Web Science Conference, WebSci 2017, Troy, NY, USA, June 25 - 28, 2017*, pages 229–233.
- [Kshirsagar et al.2018] Rohan Kshirsagar, Tyrus Cukuvac, Kathy McKeown, and Susan McGregor. 2018. Predictive embeddings for hate speech detection on twitter. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 26–32. Association for Computational Linguistics.
- [Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- [Padilla Montani and Schüller2018] Joaquin Padilla Montani and Peter Schüller. 2018. Tuwienkbs at germeval 2018: German abusive tweet detection. In *Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018)*, 09.
- [Pedregosa et al.2011] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- [Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- [Stammbach et al.2018] Dominik Stammbach, Azin Zahraei, Polina Stadnikova, and Dietrich Klakow. 2018. Offensive language detection with neural networks for germeval task 2018. In *Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018)*.

[von Grunigen et al.2018] Dirk von Grunigen, Ralf Grubenmann, Fernando Benites, Pius Von Daniken, and Mark Cieliebak. 2018. spmmmp at germeval 2018 shared task: Classification of offensive content in tweets using convolutional neural networks and gated recurrent units. In *Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018)*.

[Waseem and Hovy2016] Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *SRW@HLT-NAACL*.

[Wiedeman et al.2018] Gregor Wiedeman, Eugen Ruppert, Raghav Jindal, and Chris Biemann. 2018. Transfer learning from lda to bilstm-cnn for offensive language detection in twitter. In *Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018)*.

[Wiegand et al.2018] Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language.

German Hatespeech classification with Naive Bayes and Logistic Regression - hshl at GermEval 2019 - Task 2

Kristian Rother

Hochschule Hamm-Lippstadt
Marker Allee 76-78
59063 Hamm

kristian.rother@hshl.de

Achim Rettberg

Hochschule Hamm-Lippstadt
Marker Allee 76-78
59063 Hamm

achim.rettberg@hshl.de

Abstract

This paper describes the entries *hshl_coarse_1* and *hshl_fine_1* for Subtask I (Binary Classification) and Subtask II (Fine-grained classification) of Task 2 of the the GermEval 2019 competition. For this task, German tweets were classified as either OFFENSE or OTHER (Subtask I) or into the four subcategories PROFANITY, INSULT, ABUSE or OTHER (Subtask II). The entries employ a mixture of character level Logistic Regression and Naive Bayes. The classifiers were trained on the labeled tweets that were provided by the organizers of the shared task. The optimization of the system is outlined in this paper. The system reached an F1-score of 0.7458 and 0.4793 for the two subtasks on the test-set.

1 Introduction

Hate speech is on the rise in online communication and can come in different forms but usually follows certain patterns (Mondal et al., 2017). Additionally social media serves as a breeding ground for deviant behavior following real world incidents (Williams and Burnap, 2015).

Hate speech has psychological consequences for the victims such as fear, anger and vulnerability (Awan and Zempi, 2015) as well as the worry that online threats may become a reality (Awan and Zempi, 2016). Additionally, hate speech can be the harbinger of actual violence. Hate speech towards a group can serve as a predictor of violence towards that group (Müller and Schwarz, 2018a) and Twitter use can fuel hate-crimes (Müller and Schwarz, 2018b).

Institutions and legislators have reacted to this trend towards hate speech. The European Commission and multiple social media companies agreed to a code of conduct on countering illegal hate speech

online (European Commission, 2016). Germany passed the *Network Enforcement Act* on September 1st 2017 to enforce fines of up to 50 million Euros against social media companies that fail to delete illegal content (German Bundestag, 2017). The law specifically includes hate speech (§§130, 166 and 185-187 of the Criminal Code).

Due to the negative impact of hate speech and the amount of social media data that is generated every day, automated detection and classification of hate speech has been studied widely. Recent overviews can be found in (Schmidt and Wiegand, 2017) and (Fortuna and Nunes, 2018). However, with some exceptions such as (Ross et al., 2017) and (Van Hee et al., 2015), the scope of the studies is often limited to the English language.

As a result, the GermEval competition was launched in 2018 (Wiegand et al., 2018). By and large, (deep) neural models performed best in 2018, which is why this entry for the 2019 version of the competition focuses on a simpler, classical machine learning approach to provide a contrast to the expected neural models. Therefore, this paper tries to contribute to the improvement of the state of the art in German hate speech detection by describing the entries *hshl_coarse_1* and *hshl_coarse_2* which participated in Taks 2 at GermEval 2019.

2 Experimental Setup

The following section describes the experimental setup, namely all used technical resources, the used data and the chosen architecture.

2.1 Technical Resources

All experiments were conducted in Jupyter Notebooks, version 4.0.2 (Kluyver et al., 2016) running a Python 3.5.0 (Python Software Foundation, 2018) kernel with the following libraries:

- pandas 0.23.4 (McKinney, 2010)
- NumPy 1.11.3 (Oliphant, 2006)

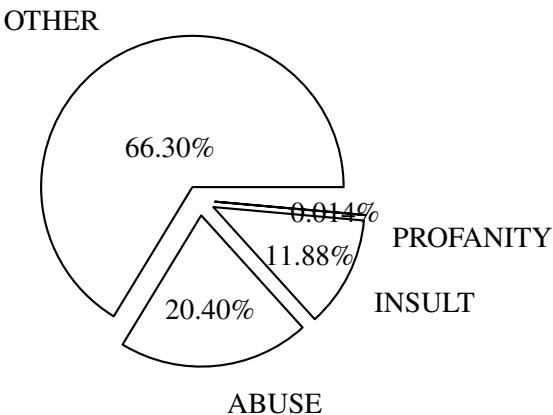
- scikit-learn 0.20 (Pedregosa et al., 2011)
- spaCy 2.0.12 (Honnibal and Montani, 2018)

A fixed seed was used for the random number generators. All models were trained on an end of 2013 MacBook Pro with a 2 GHz Intel Core i7, 8 GB 1600 MHz DDR3 and an Intel Iris Pro 1536 MB GPU¹.

2.2 Data

To train and tune the model, the data from the 2018 competition was used. This data consists of 5009 labeled tweets for the training-set and 3532 labeled tweets for the dev-set². The distribution of the classes for Subtask II is depicted in figure 1. Note that for Subtask I, the classes ABUSE, INSULT and PROFANITY are simply summarized as one class, named OFFENSE and thus this class consists of 33.70% of all tweets. The data is imbalanced. Specifically, the class OTHER makes up two thirds of the data while the class PROFANITY only represents 0.014 percent of all tweets.

Figure 1: Class distribution 2018 training-set.



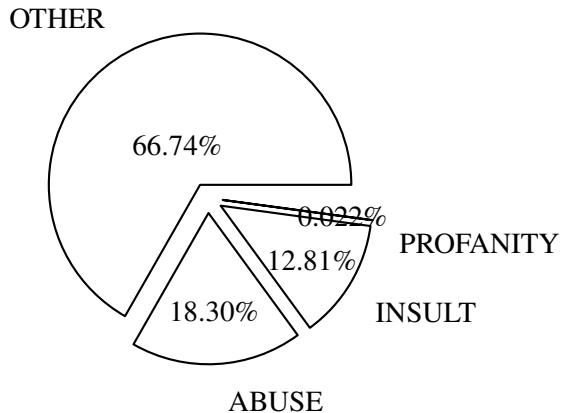
For the final submission, the system was trained on all available labeled tweets. These consist of the 2018 training- and test-set as well as the 2019 training-set for a total of 12.401 labeled tweets³. The class distributions for the 2019 training-set are described in figure 2. No additional outside data was used.

¹With this setup, the entire end-to-end training on the provided development set from 2018 took 14 seconds for Subtask A and 55 seconds for Subtask B.

²This dev-set used to be the test-set in the 2018 competition but the organizers released the labels after the competition was over.

³Some tweets were removed because they could not be parsed correctly.

Figure 2: Class distribution 2019 training-set.



2.3 Architecture

A simple bag-of-words approach that relies on the frequency of words to classify the data was used for the competition. Thus, the tweets were vectorized before they were used in the algorithms. For the vectorization, a term-frequency times inverse document-frequency matrix was used. The chosen Logistic Regression model combined with a Naive Bayes approach.

3 Experiments

The following section describes the experiments that were conducted. Because experiments could improve results for one subtask but lead to worse results on the other subtask, it was decided to use Subtask I results to pick the parameters to keep. For all experiments, spaCy was used as the tokenizer with no lemmatization and no stopword list⁴. For the term-frequency matrices, only words that appeared in at least 4 documents were used and words that appeared in more than 40% of documents were ignored. Inverse document-frequency-reweighting (IDFR) and sublinear term frequency scaling were applied, but no smoothing was applied. The parameters are summarized in table 1.

Parameter	Value
Minimum Number of Documents	4
Maximum Frequency of Documents	0.4
IDFR	Yes
Sublinear Term Frequency Scaling	Yes
Smoothing	No

Table 1: Parameters for the term-frequency matrices.

⁴As these didn't improve results in some previous tests.

3.1 Accent Stripping, Casing and Preprocessing

The first experiment was to test unicode accent stripping and lowercasing combinations. These tests were conducted for word-level unigrams. The results are summarized in table 2. Note that the last two cases are the same due to the order of lowercasing and stripping. Notably, removing lowercasing lead to worse results overall while accent stripping provided slightly better results for both subtasks.

Strip	\times	✓	\times	✓
Lowercase	✓	✓	\times	\times
F1 Sub I	0.6655	0.6665	0.6467	0.6467
F1 Sub II	0.3970	0.3975	0.3917	0.3917

Table 2: Unicode accent stripping and lowercasing. The F1 scores are micro-averaged on the dev-set.

3.2 N-grams

Different word-level n-grams were tried ⁵. Neither bigrams nor trigrams outperformed the unigrams for Subtask I. However, bigrams were best for Subtask II. The results are summarized in table 3.

In-word n-grams were also tried but didn't show any improvements.

N-gram	F1 Sub I	F1 Sub II
1,1	0.6665	0.3975
1,2	0.6496	0.4114
1,3	0.6490	0.4053
2,3	0.5874	0.3090

Table 3: Word-level n-grams. The F1 scores are micro-averaged on the dev-set.

Finally, character-level n-grams with a min to max difference of three were tried. The results are summarized in table 4.

N-gram	F1 Sub I	F1 Sub II
1,4	0.6474	0.3736
2,5	0.6672	0.3807
3,6	0.6693	0.3855

Table 4: Character-level n-grams. The F1 scores are micro-averaged on the dev-set.

⁵As a note for reproducability, it was discovered after the submission, that this experiment was run with lowercasing set to false even though the previous experiment would suggest to use true. However the final results were not impacted by this. And lowercasing was set to true for the following experiments.

The best character-level model outperformed the best word-level model for Subtask I and was thus kept for the next step.

3.3 Hyperparameters of the Logistic Regression

The hyperparameter tuning for the character-level model is summarized below. Automatically adjusting weights inversely proportional to class frequencies (balanced) provided big gains as summarized in table 5.

Balanced	\times	✓
F1 Sub I	0.6693	0.7142
F1 Sub II	0.3855	0.4514

Table 5: Balanced class weights. The F1 scores are micro-averaged on the dev-set.

Finally, different C-values for the L2 regularization were tested. The results are summarized in table 6. Because C=14 lead to the best score for Subtask I, this value was picked.

C-value	F1 Sub I	F1 Sub II
1.0	0.7142	0.4514
16.0	0.7194	0.4646
14.0	0.7202	0.4639

Table 6: L2 Regularization. The F1 scores are micro-averaged on the dev-set.

3.4 Task specific text preprocessing

As the last step, Twitter specific preprocessing was applied. The strings *RT* and 's and the symbols : and # were removed. All urls were replaced with xx_url and all @usernames were replaced with xx_username. This increased the F1-score for Subtask I to 0.7228 (and lowered the score for Subtask II to 0.4569).

3.5 Cutoff Value for Subtask I

Another parameter that can be varied is the cutoff value for Subtask I. If the prediction is equal to or more than this cutoff value, the label 'offense' is predicted and otherwise, 'other' is predicted. The logical value to use for the cutoff would be 0.5. However, experiments showed that changing this value also influences the overall result. The best empirical value of 0.485 increased the F1-score for Subtask I to 0.7258.

4 Results

After the submission deadline for the predictions, the organizers calculated various statistics on the test-set. The results for Subtask I are summarized in table 7 and the results for Subtask II are summarized in table 8. The competition was scored on the average F1-score for each task, which is highlighted in bold in the tables.

	Precision	Recall	F1-Score
OFFENSE	0.6800	0.6134	0.6450
OTHER	0.8261	0.8641	0.8447
Average	0.7530	0.7388	0.7458
	Accuracy	Correct	Total
	78.39	2376	3031

Table 7: Results for Subtask I on the 2019 test-set. F1 scores are micro-averaged.

	Precision	Recall	F1-Score
ABUSE	0.4312	0.4075	0.4190
INSULT	0.5567	0.2353	0.3308
OTHER	0.7874	0.9326	0.8538
PROFANITY	0.5000	0.0811	0.1395
Average	0.5688	0.4141	0.4793
	Accuracy	Correct	Total
	72.65	2202	3031

Table 8: Results for Subtask II on the 2019 test-set. F1 scores are micro-averaged.

5 Conclusion

The paper presented the submissions *hshl_coarse_1* and *hshl_fine_1* that were entered for the binary and fine-grained hate speech classification task of GermEval 2019. A combination of Naive Bayes and Logistic Regression was used to classify the tweets.

To reach the final model, different experiments on vectorization and preprocessing were run. Additionally, the hyperparameters of the Logistic Regression were tuned. The final model reached micro-averaged F1 scores of 0.7458 and 0.4793 on the test-set for the two subtasks.

All relevant code will be made available at one of the authors' Github repositories⁶. Additionally, lab notes of the different experiments will be added to the repository.

⁶<https://github.com/rother/germeval2019>

6 Outlook

Further improvements could be made by using an ensemble of different classifiers. Neither the vectorization nor the preprocessing or the hyperparameter search were exhaustive and can be improved upon. Using different tokenizers or other preprocessing strategies could yield better results. Theoretically, the chosen bag of words approach that only relies on word frequencies is inferior to a model that takes word order and more details into account. As such, deep neural models, specifically RNNs, are likely to provide better overall results. Lastly different sampling strategies as a measure against the imbalanced class labels would likely improve the results.

Acknowledgments

We thank the Behr-Hella Thermocontrol GmbH for supporting this research. We also thank all reviewers and the competition organizers.

References

- Imran Awan and Irene Zempi. 2015. We fear for our lives: Offline and online experiences of anti-Muslim hostility. *Report,[online] available: http://tellmamaauk.org/wp-content/uploads/resources/We% 20Fear% 20For% 20Our% 20Lives.pdf [accessed: 7 January, 2016]*.
- Imran Awan and Irene Zempi. 2016. The affinity between online and offline anti-Muslim hate crime: Dynamics and impacts. *Aggression and violent behavior*, 27:1–8.
- European Commission. 2016. Code of conduct on countering illegal hate speech online. http://ec.europa.eu/newsroom/document.cfm?doc_id=42985.
- Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Comput. Surv.*, 51(4):85:1–85:30, July.
- German Bundestag. 2017. Act to improve enforcement of the law in social networks (network enforcement act). https://www.bmJV.de/SharedDocs/Gesetzgebungsverfahren/Dokumente/NetzDG_engl.pdf?__blob=publicationFile&v=2.
- Matthew Honnibal and Ines Montani. 2018. spaCy library. <https://spacy.io>.
- Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián

- Avila, Safia Abdalla, and Carol Willing. 2016. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press.
- Wes McKinney. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.
- Mainack Mondal, Leandro Arajo Silva, and Fabrício Benevenuto. 2017. A measurement study of hate speech in social media. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media*, pages 85–94. ACM.
- Karsten Müller and Carlo Schwarz. 2018a. Fanning the Flames of Hate: Social Media and Hate Crime. CAGE Online Working Paper Series 373, Competitive Advantage in the Global Economy (CAGE).
- Karsten Müller and Carlo Schwarz. 2018b. Making America Hate Again? Twitter and Hate Crime under Trump.
- Travis E. Oliphant. 2006. *A guide to NumPy*, volume 1. Trelgol Publishing USA.
- Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830.
- Python Software Foundation. 2018. Python programming language. <https://python.org>.
- Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Watzki. 2017. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118*.
- Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.
- Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Vronique Hoste. 2015. Detection and fine-grained classification of cyberbullying events. In *International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 672–680.
- Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language, September.
- Matthew L. Williams and Pete Burnap. 2015. Cyber-hate on social media in the aftermath of Woolwich: A case study in computational criminology and big data. *British Journal of Criminology*, 56(2):211–238.

FraunhoferSIT at GermEval 2019: Can Machines Distinguish Between Offensive Language and Hate Speech? Towards a Fine-Grained Classification

Inna Vogel Roey Regev

Fraunhofer Institute SIT

Rheinstrasse 75

64295 Darmstadt

inna.vogel@sit.fraunhofer.de

Abstract

In this paper, we describe the FraunhoferSIT submission for the “*GermEval 2019 – Shared Task on the Identification of Offensive Language*”. We participated in two subtasks: task 1 is a binary classification of German tweets on the identification of offensive language. Task 2 is a fine-grained classification to distinguish between three subcategories of offensive language. Our best model is an SVM classifier based on tf-idf character n-gram features. Our submitted runs in the shared task are: *FraunhoferSIT_coarse_[1-3].txt* for task 1 and *FraunhoferSIT_fine_[1-3].txt* for task 2. Our final system reaches 0.70 macro-average F_1 -score for the binary classification and 0.46 F_1 -score for the fine-grained classification. The achieved results show that the problem of automatically distinguishing between offensive language and “Hate Speech” is far from being solved.

1 Introduction

In the pseudonymous environment of social media and due to the massive rise of user-generated content on the internet, hate speech and offensive language are easily produced and spread. As the amount of harmful comments and posts is continuously growing, it is not feasible to manually check each text message for suspicious content. Additionally, hate speech violates more than just feelings. It can be extremely harmful to society, for example by inciting mass violence. Governments and social network platforms can benefit from automatic detection and prevention of malicious posts on the net (Fortuna and Nunes, 2018).

But what constitutes hate speech and when does it differ from offensive language? A unified definition does not exist yet. The consensus is that

hate speech targets disadvantaged social groups in a manner that is potentially harmful to them (Jacobs and Potter, 2001). Many studies still tend to conflate hate speech and offensive language. A key challenge for automatic hate speech detection on social media is the separation of hate speech from other instances of offensive language (Davidson et al., 2017).

As there is a pressing demand for methods to automatically identify hate speech and other suspicious posts, we participate in this year’s “*GermEval Task 2019 – Shared Task on the Identification of Offensive Language*”¹. The task is focused on detecting offensive comments in a set of German tweets in three subtasks. Task 1 is a binary classification to distinguish offensive from non-offensive tweets. Task 2 requires a more fine-grained classification of offensive tweets which are divided into three subcategories - profanity, insult and abuse (Ruppenhofer et al., 2018). The first category “PROFANITY” uses insults and swear words, but not against a person or a group. Tweets that are labeled as “INSULT” want to offend someone. We categorize “ABUSE” as hate speech as it uses “language that is used to express hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group” and can be defined according to Davidson et al. (2017) as hate speech. Subtask 3 focuses on the classification of explicit and implicit offensive language.

We participated in task 1 and 2 of the competition. In this paper, we report on the FraunhoferSIT system to classify German tweets with respect to their offensiveness. First, we give a short overview of the work already conducted in the field of offensive language detection. In Section 3, we describe the competition tasks and the data provided by the GermEval organizers. Section 4 is dedicated to the

¹Germeval Task 2, 2019: <https://projects.fzai-h-da.de/iggsa/>

machine learning methodology used. We describe the text preprocessing and the features we used to train our SVM models. In Section 5, we evaluate the performance of our approaches. Lastly, we conclude our paper in Section 6.

2 Related Work

A wide range of machine learning and deep-learning approaches have been implemented to automatically detect offensive language in text data. Schmidt and Wiegand (2017) provide in their survey a short and comprehensive overview of automatic hate speech detection with a focus on feature extraction. Using bag of words or embeddings can yield to reasonable classification performance. In their survey, they outlined that character-level-approaches perform better than token-level-approaches because they reduce the spelling variation problem often faced when working with user-generated content. Lexical resources, such as a list of slurs, can also help to improve the performance but only if they are combined with other types of features.

Nobata et al. (2016) combine lexical features such as n-grams, as well as syntactic features with distributional semantics to detect abusive language in English comments on “Yahoo! Finance and News”. Badjatiya et al. (2017) and Davidson et al. (2017) also confirm that word n-grams are well-performing features for the detection of hate speech and abusive language.

Supervised approaches like Logistic Regression (Djuric et al., 2015; Del Vigna et al., 2017) or Support Vector Machines (Del Vigna et al., 2017; Davidson et al., 2017) have shown to obtain good results in classifying abusive language. Del Vigna et al. (2017) trained their SVM with word embeddings and Davidson et al. (2017) used bigram, unigram and trigram features, each weighted by its tf-idf. LSTM (Del Vigna et al., 2017; Badjatiya et al., 2017) and Convolutional Neural Network classifiers employed on word embeddings or other pretrained representations of words and tokens are also highly effective for the task of classifying abusive language (Badjatiya et al., 2017; Ho Park and Fung, 2017).

Most of the research is focused on English datasets. The identification of toxicity in German language messages received less attention by the researchers so far. Comparable cross-lingual research is sparse.

3 GermEval Competition Task 2019

The organizers of GermEval 2019 provided training and test datasets² for three offensive language detection tasks. The provided datasets for training consist of 12,536 tweets without any user metadata. Task 1 is a binary classification for deciding whether a German tweet contains offensive language (the category labels and the number of tweets are “OFFENSE”: 4,177 and “OTHER”: 8,359). Task 2 is a multi-class classification and distinguishes between three subcategories of offensive language with the more fine-grained labels “PROFANITY”: 271, “INSULT”: 1,601 and “ABUSE”: 2,305. While training data contains examples of all categories, the class distribution is fairly imbalanced. The majority of tweets are neutral (67%). Abusive tweets are also relatively often (18.4%), while the class “PROFANITY” is underrepresented (2.2%). The three classes that contain offensive language are defined as follows (Ruppenhofer et al., 2018):

- **Profanity:** abusive words are used but the tweet does not insult someone (e.g. *“ich scheiß auf deine Gedenkkultur sie geht mir am Arsch vorbei”*)
- **Insult:** profanity is directed at an individual with the intention to insult the person or group (e.g. *“SPD ihr seit wirklich das Asozialste Pack”*)
- **Abuse:** the tweet is intended to demean and attack another person or a group with cruel and derogatory language which we categorize as hate speech (e.g. *“Der verdammte Dreck-sack und Massenmörder Israel will mit allen Mitteln ein Krieg gegen Russland”*)
- **Other:** the tweet is neutral and does not contain any assertive or offensive words (e.g. *“Ach so vergessen, einen schönen guten Morgen”*)

Subtask 3 focuses on the classification of explicit and implicit offensive language. Explicit tweets directly express hatred towards a particular target. With implicit tweets, hatred of a target must be derived from the context. In this paper, we focus on the binary classification (task 1) as well as the more challenging fine-grained classification task 2.

²GermEval 2019 Data and Tasks: <https://projects.fzai.h-da.de/iggsa/projekt/>

The competition results are evaluated with regard to macro-average F_1 -score, which is the unweighted mean of the F_1 -scores of each individual category.

4 Methodology

In the following, the same approach is applied to both classification tasks. First, the Twitter data was preprocessed to handle idiosyncrasies such as hashtags, Emojis or irrelevant characters. Afterwards, character n-grams are extracted as features which serve as tf-idf weighted input to train a Support Vector Machine (SVM).

4.1 Preprocessing

Before extracting features and training the SVM on the tweets, we perform different pre-processing techniques according to the following procedure:

1. Removing unnecessary white spaces.
2. Lowercasing all characters.
3. Smileys and Emojis are replaced with CLDR (Common Locale Data Repository) - short character names and keywords.
4. Replacing question marks with the placeholder *<question>* and numbers with the placeholder *<number>*.
5. Deleting irrelevant symbols and characters, e.g. "+, *, /,".
6. Sequences of the same characters with a length greater than three are removed.
7. Removing words with less than three characters.
8. Removing stopwords by using the NLTK (Natural Language Toolkit) list of stopwords for the German language.
9. To tokenize the words we used the TwitterTokenizer from the NLTK library. The TwitterTokenizer is adapted for Twitter and other forms of casual speech used in social networks. It contains some regularization and normalization features (e.g. converting tweets to lowercase and vice-versa, removing username mentions and reducing the length of words in the tweet with repeated characters).

10. Hashtags (#) and the attached token were processed in a special way since they are widely used on Twitter and contain semantic content. First, the hash sign was removed. Compound words were separated in a sequence of meaningful terms (e.g. "#NoAfD" - "No AfD", "#KölnerTreff" - "Kölner Treff", "#ichwars" - "ich wars", "#OSZE-Beobachter" - "OSZE Beobachter", "#EU-Beitrittsgespräche" - "EU Beitrittsgespräche").

We also experimented with replacing all hashtags and @-Mentions with placeholders (*<HashTag>*, *<AM>*). Since this had no impact on the performance of the classifier, we discarded this preprocessing step from the final submission.

4.2 Features

After preprocessing, we used scikit-learn's³ term frequency-inverse document frequency (tf-idf) weighting function (*tfidfVectorizer*) to convert the tokens to a matrix of tf-idf features in order to build a vector pipeline (Pedregosa et al., 2011). The following n-gram features have been tested for both classifiers:

- a) word n-grams with $n \in \{1, 2, 3\}$
- b) character n-grams with $n \in \{1, 2, \dots, 7\}$
- c) Feature union of word and character n-grams

The best performance was achieved by training b). When building the vocabulary, terms that have a document frequency lower than 2 were ignored.

4.3 SVM Implementation

To train the two classifiers, we used scikit-learn's SVM with tf-idf character n-grams in the range 1 to 7 as features. Task 1 is a binary classification with the classes "OTHER" and "OFFENSE". Task 2 is a multi-class classification problem with the four classes "OTHER", "PROFANITY", "INSULT" and "ABUSE". For both models OVR ("One-vs.-Rest") was used as a decision function. To solve a multiclass classification task, OVR combines multiple binary SVM classifiers (Huang et al., 2005). Each SVM classifies samples into the corresponding class against all the other classes (Hong and Cho, 2006).

When experimenting with the training set, we split the data provided by the organizers into two

³<http://scikit-learn.org>

parts. For training, we used 70% of the data. The remaining 30% were used as test set. We experimented with hyperparameter tuning, manually and by employing scikit-learn’s grid search function. The performance improved slightly by using Sigmoid Function instead of the Radial Basis Function (RBF) kernel.

For the fine-grained classification (task 2) we initially trained two models. The first model is a binary classifier (the same approach as used for task 1) distinguishing between the two classes “OTHER” and “OFFENSE” - the latter containing all three of the abusive language classes described above. The second model was trained to differentiate between the three classes of offensive language “PROFANITY”, “INSULT” and “ABUSE”. In a second approach to task 2, a multi-class SVM was used, which was trained on all four classes simultaneously. Our second approach achieved slightly better performance results.

Finally, each model was performed on the official GermEval 2019 test set. For every task we submitted three runs, each tested with different data sets. The first run was trained on the entire dataset provided by the GermEval organizers. The second was trained only on this year’s dataset (2019), the third on text data from 2018. The best classification results in both tasks were achieved using the whole data to train the model. The classification results are provided in the following section.

5 Evaluation

In this chapter, we report on runs using our SVM models trained on the data provided by the organizers. The outputs of the SVM classifiers were submitted to the GermEval competition under the submission name **FraunhoferSIT** (see Table 1).

Submitted runs (name)	Dataset	Task
FraunhoferSIT_coarse_1.txt	2018/19	Task 1
FraunhoferSIT_coarse_2.txt	2019	Task 1
FraunhoferSIT_coarse_3.txt	2018	Task 1
FraunhoferSIT_fine_1.txt	2018/19	Task 2
FraunhoferSIT_fine_2.txt	2019	Task 2
FraunhoferSIT_fine_3.txt	2018	Task 2

Table 1: Submitted runs by FraunhoferSIT

We participated in task 1, the binary classification task distinguishing offensive from non-offensive tweets and the more challenging fine-grained clas-

sification task 2. For each task, we submitted three runs.

The GermEval task defines the macro-average F_1 -score as its evaluation measure. The results of Tasks 1 and 2 have shown that there are only minor differences in performance. Our best model, an SVM classifier, using tf-idf weighted character n-gram features on the entire training data provided by the organizers achieved the best performance among the three runs submitted for the tasks. For task 1 a macro F_1 -score of 0.70 could be achieved. Our best performance for task 2 was a F_1 -score of 0.46. In our experiments, character n-grams outperformed token n-gram features. The best n-grams at the character level range from 1 to 7. We expect our model’s performance to improve further with another set of features and more training data. As shown in our submitted runs, this would be particularly helpful for the second, fine-grained task, where our classifiers performed really poorly. The performance results for the best runs of task 1 and 2 are displayed in the following tables 2 and 3.

Task 1: SVM with tf-idf character n-grams

Category	Performance Measure		
	P	R	F_1
Other	81.24	78.62	80.42
Offense	58.46	60.93	59.67
Average	69.85	70.27	70.06

Table 2: Official evaluation results for each category of task 1 with the metrics Precision (P), Recall (R), and F_1

Task 2: SVM with tf-idf character n-grams

Category	Performance Measure		
	P	R	F_1
Other	81.37	77.58	79.43
Profanity	50	9.01	15.27
Insult	34.74	39.43	36.94
Abuse	33.52	44	38.05
Average	49.91	42.51	45.91

Table 3: Official evaluation results for each category of task 2 with the metrics Precision (P), Recall (R), and F_1

6 Conclusion

In this paper, we have described the system submitted to the “*Shared Task on Identification of Offensive Language GermEval 2019*” by FraunhoferSIT

(Darmstadt). We participated in two tasks. The first task is a binary classification of German tweets on the identification of offensive language with the two classes “OFFENSE” and “OTHER”. The second task was a more challenging fine-grained classification. In addition to detecting offensive tweets, the task was to distinguish between the three subcategories: “PROFANITY”, “INSULT” and “ABUSE”. We trained an SVM as part of scikit-learn’s library and used tf-idf character n-grams in the range 1 to 7 as features. Our model achieves a macro F_1 -score on task 1 of 0.70 and on task 2 of 0.46. The relatively poor results, especially for task 2, show that our system cannot reliably distinguish between offensive language and hate speech and that more research needs to be done to improve the classification performance.

Acknowledgments

This work was supported by the German Federal Ministry of Education and Research (BMBF) under the project “X-SONAR” (Extremist Engagement in Social Media Networks: Identifying, Analyzing and Preventing Processes of Radicalization).

References

- Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW ’17 Companion, pages 759–760, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.
- Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *ICWSM*.
- Fabio Del Vigna, Andrea Cimino, Felice Dell’Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. 01.
- Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web*, pages 29–30. ACM.
- Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Comput. Surv.*, 51(4):85:1–85:30, July.
- Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. 06.
- Jin-Hyuk Hong and Sung-Bae Cho. 2006. Multi-class cancer classification with ovr-support vector machines selected by naïve bayes classifier. In Irwin King, Jun Wang, Lai-Wan Chan, and DeLiang Wang, editors, *Neural Information Processing*, pages 155–164, Berlin, Heidelberg. Springer Berlin Heidelberg.
- D.S. Huang, X.P. Zhang, and G.B. Huang. 2005. *Advances in Intelligent Computing: International Conference on Intelligent Computing, ICIC 2005, Hefei, China, August 23-26, 2005, Proceedings*. Number Teil 1 in Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- James B. Jacobs and Kimberly Potter. 2001. *Hate crimes: criminal law and identity politics*. Oxford University Press New York.
- Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW ’16, pages 145–153, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, November.
- Josef Ruppenhofer, Melanie Siegel, and Michael Wiegand. 2018. Guidelines for iggsa shared task on the identification of offensive language. *March*, 12:2018.
- Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

FoSIL - Offensive language classification of German tweets combining SVMs and deep learning techniques

Florian Schmid, Justine Thielemann, Anna Mantwill, Jian Xi, Dirk Labudde, Michael Spranger
 University of Applied Sciences Mittweida

Technikumplatz 17

09648 Mittweida

spranger@hs-mittweida.de

Abstract

In this paper an approach for the automatic detection of offensive language in German twitter posts, so called tweets, based on a data set provided by the organizers from the GermEval2019 contest is presented. Two different approaches were used. The first one is based on a document-term-matrix and the second one uses fastText to represent tweets as numerical vectors. Additionally, some text based features, e.g. sentiment analysis of the text and emojis were added. Further, some statistic features were calculated, e.g. the number of special characters, hashtags and mentions. As a classifier a support vector machine with radial kernel function was utilized. The best f1-macro values for subtask 1 of 0.7978, subtask 2 of 0.5957 and for subtask 3 of 0.7055, validated by a ten-fold cross validation, were achieved by using a self-trained unsupervised fastText model to vectorize the tweets.

1 Introduction

Social media platforms like Twitter have become increasingly popular in the past ten years (Twitter, 2019). People of nearly all generations, especially teenagers and young adults, are using them to communicate with friends, connect with people around the world or to state their opinion about current topics (Faktenkontor, 2019). Unfortunately, the increasing number of people using social media platforms results in a growth of posts with offensive content. Therefore, the automatic detection of offensive language on these platforms is a very important task to effectively fight e.g. hate speech, hateful or insulting comments, cyber mobbing or cyber bullying.

The detection of offensive language is a typical task in sentiment analysis, which is in turn a sub-

task in text classification, that focuses on the contextual mining of texts related to some specific objects. Furthermore, sentiment analysis is especially useful to find out the public opinion concerning highly sensitive political topics, as was shown in the study by Backfried et al. (2016), in which Twitter texts were analysed in order to detect tendencies that are inter-related to real world events in the European refugee crisis. Usually, sentiment analysis involves methods from different disciplines such as natural language processing and machine learning (Pang et al., 2002).

The challenge by the organizers of the GermEval 2019 Task 2 focuses on detecting offensive language in tweets and is subdivided into three smaller tasks: The first task is to detect texts containing offensive language in Twitter messages. The second task is the fine-grained categorization of tweets into one of the categories neutral, profanity, insult or abuse. Finally, the third task is to distinguish offensive tweets to be explicit or implicit.

In this paper, for the first subtask two systems were used and compared. The first system uses SVM with a radial kernel function as classifier incorporating different lexical resources. This approach forms the baseline. The second system extends the first one by vectorizing the data with a self-trained fastText model based on nearly 30 million tweets. Due to better results being achieved with the second system, it was used for the other two subtasks.

The paper is organized as follows: in Section 2 an overview of the data is given. In Section 3 the methods used are described and in Section 5 the results are presented. Finally, in Section 6 a short conclusion is given.

2 Data

The data for all subtasks consisted of tweets provided by the organizers of the GermEval 2019 Task 2. For the first two subtasks the dataset con-

tained 12,536 manually labelled tweets. As can be seen in Table 1, the dataset was highly imbalanced. There is double the amount of tweets in the category OTHER compared to the category OFFENSE and even for the fine-grained classification task the number of tweets in each category varies greatly.

For the third subtask an additional dataset was provided, consisting of only 1958 tweets. Again, the dataset was imbalanced (see Table 1), with the category EXPLICIT having more than five times as many tweets as the category IMPLICIT. The tweet’s content was neither preprocessed nor cleaned and therefore contained hashtags, user mentions, emoticons and other text patterns that are typical for social media platforms (GermEval, 2019).

subtask	category	# tweets
Subtask 1	OFFENSE	4177
	OTHER	8359
Subtask 2	ABUSE	2305
	INSULT	1601
	PROFANITY	271
	OTHER	8359
Subtask 3	EXPLICIT	1699
	IMPLICIT	259

Table 1: Number of tweets in each category.

3 Methods

In this paper, two different systems are presented for the classification, each based on a SVM with a radial kernel function and the preprocessed tweets as described in the following Section 3.1. For the first system a document-term-matrix (DTM) built on a pruned vocabulary was used that holds the following condition: $1 \leq tf(w) \leq 50$, where $tf(w)$ is the term frequency of each single word from the preprocessed tweets. Further statistical features, sentiment scores and lexical resources were used as additional features. In contrast, in the second system the preprocessed tweets were vectorized using a self-trained unsupervised fastText model.

Some more detailed information is given in the following subsections.

3.1 Preprocessing

Before any further steps were taken to normalize the tweets some statistical features were calculated. An overview is given in Table 2. Afterwards, the

tweets were changed to lower case, all special German characters were converted, the punctuation marks removed and the words lemmatized using TreeTagger (Schmid, 1995).

Feature	values
tweets containing emojis	1011
tweets containing hashtags	2355
tweets containing mentions	12,536
average no. of words per tweet	18.22
average no. of punct. marks per tweet	6.44

Table 2: Statistical features for both datasets.

Because hashtags are potentially important to capture the real message or sentiment of a tweet, only the #‐sign at the beginning of a hashtag was removed, yet the hashtag itself was kept as part of the tweet.

As no further information about users or groups was given, the mentions in all tweets were removed completely. Moreover, stop words were removed using the list provided by Diaz (2016). However, this list was modified, because some stop words may give important information regarding the sentiment of a tweet. For instance, it makes a huge difference whether an adjective is preceded by a negation word or not. Furthermore, personal or possessive pronouns may indicate that someone is addressed personally. Consequently, negation words as well as personal and possessive pronouns were not removed. Finally, a document-term-matrix was created.

3.2 Feature Modelling

Sentiment Analysis on Texts and Emojis

To get the sentiment of a tweet, a combined score was calculated from the words and emojis in the tweet. In order to get a sentiment score for the words SentiWS (Remus et al., 2010) was used to assign a positive or negative polarity value between -1 and 1 to each word. Emojis were taken into account, because, usually, a large number of tweets contain emojis (Gotzner, 2013) and because, in some cases they can indicate the mood or clarify the meaning of an expression. In order to calculate a sentiment score for the them, the Emoji Sentiment Ranking (Kralj Novak et al., 2015) from the Department of Knowledge in Slovenia was used. First, the emojis were extracted, converted to their unicode sequence (e.g. <U+263>) and then a score between -1 and 1 was assigned.

Finally, the single scores were summed up for each tweet.

Lexical Lookup

Due to the young age of twitter users, colloquial words or teenage-slang-words are often included in tweets. Several teenage-slang-words are offending either a single individual or groups of them. To detect these words a lexicon of youth language was used, which was created by Helmut Hehl (Hehl, 2006). However, some phrases were removed manually because they were not relevant for detecting offensive language.

Additionally, in order to detect swearwords in tweets a comprehensive lexicon containing offensive nouns, adjectives and also verbs was created. The nouns were obtained from the “HyperHero Schimpfwortliste”, a huge list with 11,300 swearwords (HyperHero, nd). The adjectives with an abusive connotation were manually extracted from the website www.wortwuchs.net (Willing and Goldschläger, nd). The verbs were added manually because there was no suitable list available. Finally, some words, which were significant for the data were added manually in their lemmatized form and all lists were combined to our comprehensive lexicon. For each tweet, a binary decision was made whether the tweet contains offensive or slang words from our lexicons or not.

Vectorization

The language used in social media is strongly related to currently discussed topics. Therefore, each sample drawn from social media captures only a limited amount of the vocabulary used. To overcome this limitation, a huge amount of tweets were collected to capture as much of the vocabulary as possible incorporating different topics in order to build a fastText model (Joulin et al., 2017).

FastText is able to capture the context of words instead of simply checking if a word is in a tweet or not. In order to train a suitable fastText model a crawler was set up that automatically collects German tweets from the twitter API. This way, an additional dataset was created consisting of finally 29.6 million unique German tweets each preprocessed as described in Subsection 3.1. This slowly growing corpus formed the basis for training different unsupervised fastText models to subsequently create a vectorized text representation of the data provided by the organizers. At different points in time models were created in order to analyse the per-

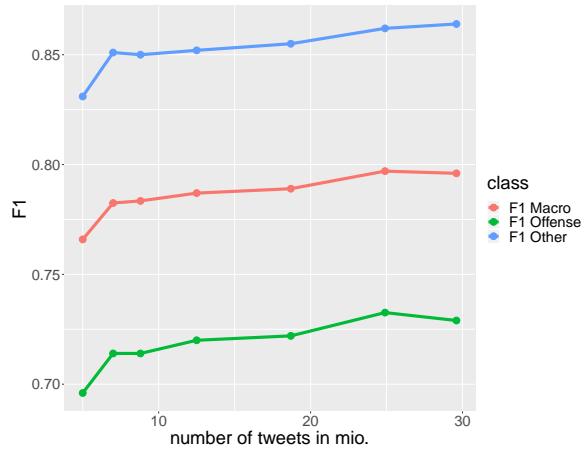


Figure 1: The plot shows the overall performance of system 2 with increasing numbers of tweets used to train the fastText model.

formance depending on the number of tweets used for the fastText model. As can be seen in Figure 1 with a growing number of tweets the results also increase until the number of tweets reaches 24.9 million. Afterwards, the macro F1 measure slightly decreases. For the two submitted runs the decision was made to use the fastText model that achieved the best result in the ten-fold cross-validation with 24.9 million unique German tweets and the fastText model with the final amount of 29.6 million tweets.

As parameters 50 epochs, 300 dimensions, a window size of 5, char N-grams with a length from 2 to 6 and a learning rate of 0.05 were used. The usage of char N-grams make the model more robust against unseen words. The models were calculated using a continuous-bag-of-words and skip-gram technique as well as a hierarchical softmax function and negative sampling.

4 System Descriptions

In this paper, two different systems were used for the classification. In the following the different systems are described.

System 1 - DTM and SVM (radial kernel)

The first system is based on the preprocessed tweets and a document-term-matrix (DTM) which was built with the pruned vocabulary from the training data set ($\text{min tf} = 1$, $\text{max tf} = 50$). Additionally, some statistical features, sentiment scores and lexical resources were added. As a classifier a support vector machine with a radial kernel function was used.

System 2 - fastText and SVM (radial kernel)

The second system is based on the preprocessed tweets which were vectorized by a self trained unsupervised fastText model. As for the first system, statistical features, sentiment scores and lexical resources were added. Again, a support vector machine with a radial kernel function was used. For the 1st run a fastText model built on 24.9 million unique tweets was used, whereas for the 2nd run a fastText model built on 29.6 million unique tweets was used.

5 Results

The following tables show the results for each subtask and system. All results are based on the training data set and a ten-fold cross-validation to prevent overfitting of our models.

The results in Table 3 show the best achieved scores with system 1, which represent the start of development. This system formed the baseline for the further work and results were not submitted to the contest.

Run	Category	P	R	F1
-	OFF.	0.5640	0.4096	0.4742
-	OTHER	0.7405	0.8415	0.7877
	Mac. avg.	0.6522	0.6255	0.6386

Table 3: Results for subtask 1 with system 1 (not submitted).

The following Tables 4 to 6 show the best scores achieved with system 2 and the different runs as described in Section 4. Both runs were submitted to the contest.

Run	Category	P	R	F1
1 st	OFF.	0.7193	0.7467	0.7326
	OTHER	0.8710	0.8543	0.8625
	Mac. avg.	0.7952	0.8005	0.7978
2 nd	OFF.	0.7291	0.7302	0.7291
	OTHER	0.8650	0.8637	0.8643
	Mac. avg.	0.7967	0.9770	0.7968

Table 4: Results for subtask 1 with system 2.

With the first model, several problems occurred. The large vocabulary of more than 22,000 unique words led to a high sparsity of the DTM, which

Run	Category	P	R	F1
1 st	ABUSE	0.5567	0.6130	0.5822
	INSULT	0.4875	0.4866	0.4865
	PROF.	0.6669	0.2839	0.3950
	OTHER	0.8602	0.8526	0.8563
	Mac. avg.	0.6428	0.5586	0.5975
2 nd	ABUSE	0.5372	0.6239	0.5769
	INSULT	0.4629	0.5391	0.4978
	PROF.	0.5851	0.3134	0.4033
	OTHER	0.8739	0.8200	0.8460
	Mac. avg.	0.6148	0.5741	0.5935

Table 5: Results for subtask 2 with system 2.

Run	Category	P	R	F1
1 st	IMPLIC.	0.3582	0.6678	0.4653
	EXPLIC.	0.9418	0.8164	0.8744
	Mac. avg.	0.6500	0.7421	0.6929
2 nd	IMPLIC.	0.3660	0.7143	0.4825
	EXPLIC.	0.9489	0.8081	0.8725
	Mac. avg.	0.6575	0.7612	0.7055

Table 6: Results for subtask 3 with system 2.

in turn caused different computational problems. Therefore, the vocabulary was pruned, as described in the former section, in order to reduce its size to around 1,100 words. However, pruning the vocabulary also means that a lot of information from the tweets gets lost. As the results in Table 3 clearly show, with the first system it was not possible to detect much of the offensive language in the dataset. As can be seen in Table 4 the results for subtask one clearly improved using the fastText model. As might be expected, the results are worse for the second subtask (see Table 5). The results clearly show that it is really difficult to detect profanity in the tweets, whereas for the category ABUSE the best results were achieved. However, the results also coincide with the number of tweets available. For PROFANITY the number of tweets was the lowest, while for ABUSE it was much higher. Furthermore, the results in Table 6 indicate that it is more difficult to detect implicit abusive language in comparison to explicit abusive language. Yet again, the bad results can be partly explained with the available number of tweets. Interestingly, a greater number of tweets for the training of the fastText

model does not improve the results for the first two subtasks. However, the difference between the two runs is minimal for all three subtasks.

6 Conclusion

As pointed out in the discussion section, it can be clearly seen how modern techniques for word representations like fastText can help achieve better results in natural language processing tasks. Using a radial SVM and a fastText vectorization as a feature, for the first subtask an F1-measure of 0.7978 was achieved, whereas for the second and third subtask the F1-measure was 0.5975 and 0.7055, respectively.

The deep learning technology used for fastText enables the transformation of most of the context into numerical vectors with a moderate number of dimensions. This led to an increase of the overall performance of our model in the second system. Besides fastText there are many different implementations for modern word embeddings like word2vec, sent2vec or doc2vec. It might be interesting to use different word embedding techniques for the text vectorization as well as classifier chains.

References

- Gerhard Backfried and Gayane Shalunts. 2016. Sentiment analysis of media in german on the refugee crisis in europe. In Paloma Díaz, Narjès Bellamine Ben Saoud, Julie Dugdale, and Chihab Hanachi, editors, *Information Systems for Crisis Response and Management in Mediterranean Countries*, pages 234–241, Cham. Springer International Publishing.
- Gene Diaz. 2016. Collection of Stopwords for multiple languages. <https://github.com/stopwords-iso/stopwords-iso>. Accessed: 27.03.2019.
- Faktenkontor. 2019. Anteil der befragten Internetnutzer, die Twitter nutzen, nach Altersgruppen in Deutschland im Jahr 2017. Statista. <https://de.statista.com/statistik/daten/studie/691593/umfrage/anteil-der-nutzer-von-twitter-nach-alter-in-deutschland/>. Accessed: 24.07.2019.
- GermEval. 2019. Germeval Task 2, 2019 — Shared Task on the Identification of Offensive Language. <https://projects.fzaih.de/iggsa/germeval/>. Accessed: 01.07.2019.
- Peter Gotzner. 2013. Herzchenzähler analysiert das Twitter-Gefühlsleben. Accessed: 24.07.2019.
- Helmut Hehl. 2006. Lexikon der Jugendsprache. Accessed: 14.06.2019.
- HyperHero. n.d. HyperHero Schimpfwortliste. <http://www.hyperhero.com/de/insults.htm>. Accessed: 03.04.2019.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 2, pages 427–431.
- Petra Kralj Novak, Jasmina Smajlović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of Emojis. *PLOS ONE*, 10(12):1–22, 12.
- Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86.
- Robert Remus, Uwe Quasthoff, and Gerhard Heyer. 2010. SentiWS - a publicly available German-language resource for sentiment analysis. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Languages Resources Association (ELRA).
- Helmut Schmid. 1995. Improvements In Part-of-Speech Tagging With an Application To German. In *In Proceedings of the ACL SIGDAT-Workshop*, pages 47–50.
- Twitter. 2019. Anzahl der monatlich aktiven Nutzer von Twitter weltweit vom 1. Quartal 2010 bis zum 1. Quartal 2019 (in Millionen) [Graph]. Statista. <https://de.statista.com/statistik/daten/studie/232401/umfrage/monatlich-aktive-nutzer-von-twitter-weltweit-zeitreihe/>. Accessed: 24.07.2019.
- Rebekka Willing and Jonas Goldschläger. n.d. Wortwuchs Adjektivliste. <https://wortwuchs.net/adjektivliste/>. Accessed: 14.06.2019.

2019 GermEval Shared Task on Offensive Tweet Detection

`h_da` submission

Isabell Börner^{*} Midhad Blazevic^{*} Maximilian Komander^{*} Margot Mieskes[†]

University of Applied Sciences, Darmstadt, Germany

^{*}firstname.lastname@stud.h-da.de

[†]firstname.lastname@h-da.de

Abstract

This paper presents the models submitted to the 2019 GermEval Shared Task on Offensive Language Detection in Tweets. Our system is based on a lexicon of swear words and several rules. These rules were developed after a thorough data and error analysis. This also revealed that the detection of offensive language is far from trivial and in a lot of cases requires more than just a Tweet in isolation, but rather would require more knowledge about the context and/or the topic the Tweet is related to, which was not available in this data set.

1 Introduction

“Offensive language is commonly defined as hurtful, derogatory or obscene comments made by one person to another person. This type of language can be increasingly found on the web.” With these words Wiegand et al. (2018) introduced the 2018 edition of the GermEval 2018 Shared Task on the identification of offensive language. While this indicates an academic interest in the topic, the German Netzdurchsetzungsgesetz (NetzDG) requires social networks to remove illegal content (Smedt and Jaki, 2018) which might overlap with offensive language in general. Recent events surrounding the murder of a German politician in June 2019, police forces look into social media containing hate speech (German “Hasskommentare”) related to this event.¹ Additionally, there is very little work on German hate speech, as opposed to English hate speech and/or offensive language.

This paper presents the description of the system submitted by the University of Applied Sciences, Darmstadt (`h_da`) to the GermEval 2019 edition

of the shared task on detecting offensive language in Tweets. While most systems in the 2018 edition used machine or deep learning, we created a rule-based system after performing a thorough data analysis.² While a range of our observations could be translated into features for machine learning, this was not the main focus of this work. Similar to (Klenner, 2018) we observe that the annotations are not as clear, as the annotations suggest. Accordingly, we feel (similar to (Smedt and Jaki, 2018)) that releasing AI without a proper verification is ethically critical. Therefore, we suggest to use confidence scores, rather than absolute annotations to indicate the potential label and to also have a closer look at the manual annotations, which are not always as clear-cut as they might seem. Especially in isolation not all annotations are comprehensible and might need some further discussion.³

2 Data Analysis

Initially, we thoroughly looked at the 2018 and 2019 data sets in order to gain a better intuition for the material we are dealing with. It became obvious, that many offensive tweets have one common ground: they use offensive language to offend certain people, institutions, countries or companies. Our idea was, that a script could classify tweets by looking for “bad” language inside the tweets and thus categorize them as either OFFENSIVE or OTHER. The basis for what we consider bad language, is a list of words found at: <http://www.insult.wiki/wiki/Schimpfwort-Liste>. Our error analysis revealed that the classification contained too many mistakes. We therefore removed words such as “Ameise” (ant), “Vielflieger” (frequent flyer) or “Bär” (bear) which do not have negative connota-

²Details of our system are available at <https://github.com/mieskes/germEval2019>

³We present examples taken from the data in German and provide a rough translation into English.

¹<https://www.zeit.de/gesellschaft/zeitgeschehen/2019-08/walter-luebcke-hasskommentare-internet>

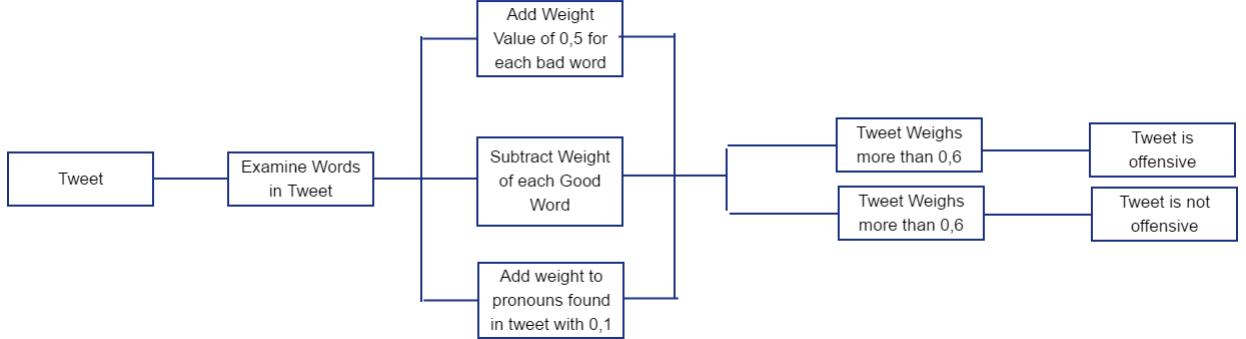


Figure 1: Architecture of the h_da lexicon- and rule-based system.

tions.

In another step, we performed an error analysis, looking in detail at the mis-classified Tweets. Errors came in two forms: One set of errors is based on the mis-classification of our method. Another set of errors can be attributed to annotations that are less clear-cut. For example, there was a tweet in which one person stated, that he wishes the old german anthem back and also the time (WW2-era), too. This tweet has not been officially classified as OFFENSIVE, while we came to the conclusion that it was indeed, offensive. This naturally leads to problems, as our script classified some tweets as OFFENSIVE because it contained those offensive words and insults, while the official file marked them as OTHER. To increase the accuracy, we looked up the tweets and gathered offensive words that our own list did not contain at this time.

As the accuracy did not increase significantly, we decided to add weights to the words in our lists. In addition, we observed that Tweets contained words which might not be offensive as such (i.e. "Hund" (english: dog), but changes to being offensive if combined with a pronoun and/or an (offensive) adjective. A sentence such as "Ein hässlicher Hund" (An ugly dog) becomes offensive in the case of "Du hässlicher Hund" (You ugly dog). We therefore added weights based on a word being in the word list, occurring with a pronoun and with an adjective.

3 Experimental Setup

The first phase consisted of using a "badword list" to identify tweets that are offensive. Our system compares the words in a tweet to the words that can be found in the "badword list", and if a tweet has one of these badwords then is considered of-

fensive. This simple comparison provided mixed results due to the "badword list" not being optimal, due to words within the list that might or might not be considered as bad or offensive words, depending on context. The second phase (shown in Figure 1 above) involved adding weights to identify offensive tweets, by analyzing as many words of the tweet as possible and using the end weight to identify if a tweet is offensive or not. The "badword list" from the first phase is being used, and all of the words that can be found in the list are weighted as +0,5. Pronouns are also weighted due to the fact that many hate speech tweets consist of, for example, a person being attacked directly by using the word "du" (you). Pronouns are currently in this stage weighted at +0,1.⁴ In order to further enhance the analysis, we used SentiWS_v2.0 (Remus et al., 2010) lists to optimize the analysis by also using positive words, along with SentiWS' value of these positive words to minimize the weights of the tweets. Furthermore, we thought of using the negative word list as well, but it misses swear words. Our own "badword list" is also being further developed. In version 2.0 the list will be newly created by real people via Google Survey, which has been sent to different people from all ages and sexes. The overall weighting system will also be fitted later on, as we proceed.

4 Results

Our system is primarily based on the list of insults as described in Section 3 above. The model looks for every bad word in the selected tweet and thus makes an assumption about its polarity. Evaluating the first runs, we notice that the classification was

⁴An experimental analysis of the weights was not possible due to time constraints.

system	Average			Offense			Other		
	p	r	f	p	r	f	p	r	f
test 2018 weighted	50.12	50.13	50.12	34.16	43.34	38.21	66.07	56.91	61.15
test 2018 unweighted	48.22	48.45	48.33	31.42	24.54	27.56	65.02	72.36	68.49
train 2019 weighted	54.97	54.63	54.35	36.64	51.98	42.98	71.51	57.27	63.60
train 2019 unweighted	57.55	55.87	56.70	44.05	29.60	35.41	71.05	82.13	76.19
final	50.12	50.13	50.12	34.16	43.34	38.21	66.07	56.91	61.15
test 2019 (official; run 1)	59.60	58.24	58.91	46.42	36.08	40.60	72.77	80.40	76.39
test 2019 (official; run 2)	54.55	58.24	54.87	36.95	52.68	43.43	72.15	57.69	64.11
post-evaluation	58.09	56.39	57.23	44.81	30.85	36.54	71.37	81.94	76.29

Table 1: Results for various variants of our system.

prone to mistakes, as our badwordlist contained too many insults and slurs that on the other hand were used in non-offensive tweets and thus resulting in false positives, with an accuracy under 50 %. We therefore reduced the amount of bad words in our list from about 2000 to 1520 to increase accuracy.

Based on our error analysis (described in Section 2) we add weights to the bad words and pronouns. A bad word receives a weight of 0.5 and selected pronouns a weight of 0.1. If the tweet has a weight of at least 0.6 it is considered offensive. The pronouns we include are "ihr", "du", "sie", "dich", "euer", "ihrer", "deren" and "dein".

Table 1 shows the results of our systems on various data sets including the official test evaluation results. We observe that the weighted system consistently has higher Recall results when labelling a Tweet as OFFENSE, whereas it achieves higher Precision when labelling a Tweet as OTHER. The unweighted model shows higher Precision for OFFENSE and higher Recall for OTHER. As recognizing an offensive Tweet is a critical task, from several points of view, it is desirable to achieve a higher Recall in order to ensure that a Tweet labelled as offensive is actually offensive.

We also combined the two models during the post-evaluation analysis, which increased the performance on average and also in both categories. The combined model takes the output of both models. In case the models agreed the decision was used. For non-anomous decisions the weighted model decided for the OFFENSE category and the unweighted model for the OTHER category.

5 Error Analysis

After the gold labels for the test data were released, we performed a detailed round of error analysis on the actual test data. The tweets themselves prove

to be a challenge. Many tweet labels are not clear, and thus even though a tweet is labeled as offensive or abusive, we do not consider every offensive labeled tweet to be offensive. We have found tweets in which a simple figure of speech such as "Ich glaub ich muss kotzen" (I think, I have to throw up) is considered offensive. Our system also found these tweets to not be offensive, and this is in our opinion correct. Another example: a tweet has been marked as OFFENSE INSULT with the content "Diese Studenten, die ihren Studienausweis zücken, bevor der Kontrolleur kommt" (Those students, who take out their student id before an inspector shows up), which in our opinion does not represent Hate-Speech at all. While for these cases contexts can be imagined, where such an utterance could be considered hate speech, others, such as "Seit wann magst du Kartoffeln?" (Since when do you like potatoes?) or "Bratkartoffeln aus rohen Kartoffeln best, aber verdammt immer eine Riesen-sauerei" (Hash Browns out of raw potatoes are the best, but that sure means a big mess) it is harder to imagine a context where these utterances could be considered offensive.

Even more challenging is how labeling occurred when looking closer at tweets that can be considered political statements, in which no person or entity is directly harmed. Some tweets can even be considered sarcastic with reference to the past. These sarcastic comments are not positive but also do not attack a person directly. Also, it is still an open question whether sarcastic or ironic comments are necessarily considered Hate Speech, as in the context of political comedy these methods are frequently used. But other tweets that are targeted towards specific groups, have not been labelled as offensive, while we came to the conclusion that they could probably be considered offensive, such

as “Klar. Danach kannste dir direkt umsonst auch noch Schläge abholen” (Sure. Afterwards you can get a beating for free).

The context behind the tweet, which is missing in the labeled data, is nonexistent for us, and for our system. This leads to problems such as incorrect labeling by the system. A tweet that is labeled by the annotators as offensive due to context, cannot always be labeled as offensive by the system. Since a simple application cannot dive deeper into context, many tweets could not be analyzed correctly.

During the manual inspection of our systems results, we decided to add words such as “Jude” (jude), “Moslem” (muslim), etc. to the first list, due to our system missing offensive labeled tweets which had these words. These words are regrettably misused for offensive purposes. Using these words in our badword list does create false positives but improves results. Examples, where we found that a tweet was not labelled as offensive, but could be considered offensive towards muslims is a tweet like: “Hey, das war ausschließlich gegen Muslime gerichtet, halb so wild!” (Hej, this was only targeted towards Muslims, no big deal!).

Using a badword list for comparison and identifying bad or offensive tweets has also proven to be difficult. We have tested our system with two different lists. The first list consists of 1.520 words. The second list consists of 11.303 words that can be used as offensive words. The overall results using the first list were better than when using the second list. The second list seemed to have falsely labeled too many tweets as offensive. Overall, we consider smaller lists that have good quality to be better than extensive list, thus quality goes over quantity.

6 Discussion & Conclusions

While our system does not outperform the others, we think that the analyses we carried out during the project are quite valuable. Additionally, these analyses indicate, that the classification of Tweets at least in most cases requires contextual and/or meta information. There are a range of cases, where it is easy to imagine, that a context might exist, which renders a Tweet harmless or harmful. Without information about previous Tweets, the topic, the Tweet under consideration refers to, it is hard to be absolutely sure.

Nevertheless, we see a range of options to improve our system. One of the first steps is, rather than relying on fixed sets of words, such as the list

of pronouns, some more linguistic preprocessing, such as Part-of-Speech tagging might prove useful. Additionally, our findings could be incorporated in a Machine Learning setup, which would benefit the overall precision/recall values.

Also, the definition of hate speech was in some cases quite strict. Several tweets have been officially classified as OFFENSE although no hate speech or offensive language could be detected by our group. We do not consider simple sarcasm or irony as hate-speech, which also results in lower accuracy rates.

On a more general note, the task of identifying offensive language has to walk a very fine line between targetting offensive language, which might also be illegal, as in the case of the German “Volksverhetzung” (incitement of the people) and censorship. Thus, from ethical point of view, we should be careful about how strict our definition of offensive language is and what has to be accepted under the freedom of speech.

References

- Manfred Klenner. 2018. Offensive language without offensive words (olwow). In *Proceedings of the GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018) Vienna, Austria, September 21, 2018*, pages 11–15.
- R. Remus, U. Quasthoff, and G. Heyer. 2010. SentiWS - a Publicly Available German-language Resource for Sentiment Analysis. In *Proceedings of the 7th International Language Resources and Evaluation (LREC’10)*, pages 1168–1171.
- Tom De Smedt and Sylvia Jaki. 2018. Challenges of automatically detecting offensive language online: Participation paper for the germeval shared task 2018 (haua). In *Proceedings of the GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018) Vienna, Austria, September 21, 2018*, pages 11–15.
- Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language. In *Proceedings of the GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018) Vienna, Austria, September 21, 2018*, pages 1–10.

HAU at the GermEval 2019 Shared Task on the Identification of Offensive Language in Microposts: System Description of Word List, Statistical and Hybrid Approaches

Johannes Schäfer¹, Tom De Smedt², and Sylvia Jaki³

¹ Institute for Information Science and Natural Language Processing, University of Hildesheim

² Computational Linguistics Research Group, University of Antwerp

³ Department of Translation and Specialized Communication, University of Hildesheim

johannes.schaefer@uni-hildesheim.de, tom.desmedt@uantwerpen.be,
jakisy@uni-hildesheim.de

Abstract

This paper presents our contribution (HAU) for the three subtasks of GermEval 2019 Task 2. To detect offensive microposts, we have experimented with different approaches and a combination thereof, namely, a Convolutional Neural Network (CNN), a Random Forest, and a lexicon-based approach. In this paper, we report our methodology, demonstrate how it includes insights from GermEval 2018, and compare the different approaches for the different subtasks in view of future directions in the detection of offensive language and hate speech online.

1 Introduction

This year’s *Terrorism Situation and Trend Report* from Europol (2019) again notes the increase of far-right incidents in Germany, including attacks against immigrants and mosques, the rioting in Chemnitz, the fatal stabbing of a pro-migrant politician, and in particular mentions the instigators’ affinity with weapons. It is becoming difficult to ignore the role of unmoderated social media platforms like *Gab*, *4chan* and *8chan* in radicalization processes and the proliferation of hatred. This in itself justifies organizing a second GermEval task on offensive language (or other shared tasks such as OffensEval 2019 (Zampieri et al., 2019) and HatEval 2019 (Basile et al., 2019)), encouraging the scientific community to investigate hateful online discourse, for example to help develop better content moderation tools, or early warning systems for illegal content.

HAU1 is our best submission: a CNN with extensive feature engineering. HAU2 is a lexicon

of offensive words and manually-annotated word scores; our main interest here is to observe how well it does compared to other techniques. HAU3 is a Random Forest trained on character trigrams and word unigrams, which we used as a baseline for the other two. The choice of the techniques is based on last year’s GermEval, where CNNs and Random Forests were among the best-performing systems (see Wiegand et al., 2018), and where we also demonstrated that competitive results can be achieved with lexicons.

In the following sections, we present our three approaches, with the main emphasis on the CNN in Section 2. The Random Forest is discussed in Section 3, and our new ‘POW’ lexicon in Section 4, followed by a discussion of the systems’ performance in the three subtasks in Section 5.

2 HAU1: CNN

In this section, we describe our neural network model developed for the identification of offensive language in microposts. The overall network structure is based on the approach described in detail in Schäfer (2018) where text, metadata and linguistic features are handled by parallel sub-networks. We use a variant that yielded the best results in previous experiments, and which considers only text-based and metadata features. We elaborate on a few choices for the structure and hyperparametrization in Section 2.3, based on early experiments on the given GermEval 2019 training dataset.

2.1 Model Description

In the following, we list the differences of our model in comparison to the approach described in Schäfer (2018), firstly, the changes to the model

architecture, and, subsequently, the changes to the training procedure.

Model architecture: The original system used a recurrent neural network (RNN) with long short-term memory (LSTM) units for encoding the text-based features. However, during experiments we observed an improved performance when replacing this encoder by a CNN. Our new CNN model is based on the architecture given in Schäfer and Burtnshaw (2019), where multiple sequences of convolution, dropout and max pooling are combined in parallel in a text encoder. This encoder operates on a word embedding of the input text sequence (both in GermEval 2018 and 2019). The parallel structure was specifically designed for the detection of offensive language. Each branch in the CNN is trained to identify those n-grams (with n from 1 to 6) from the text which are significant for the classification task. Based on that, different types of offensive expressions can be captured and also mentions of targets (proper names) or other multi-word features.

The sub-network with metadata features has remained unchanged in comparison to Schäfer (2018). It considers a variety of numerical features calculated on the input text, by using rule-based formulas and partially counting matches in pre-defined word lists. The list of 27 basic metadata features¹ is given in the referenced paper.

Model training procedure: We briefly mention two commonly used methods for training a neural network, which were not considered in Schäfer (2018) but included in the system at hand. First, the label ratio in the given dataset is imbalanced: about 1:2 in the binary classification task, or 1 offensive comment per 2 non-offensive. When trying to optimize all labels equally (which is considered by the macro-average F1-score evaluation metric of the shared task), it is beneficial to use class weights during the training process. By doing so, we can, for example, boost the importance of the more infrequent label OFFENSE in the binary classification task. We give our exact class weights used for the submitted system runs for the different tasks computed on the entire training dataset in Section 2.3.

Second, as we train the model by iterating over the training data multiple times, a stopping point has to be determined, early enough to avoid overfitting to the given training data. To automatically

determine a suitable number of training epochs, we use early stopping as follows. About 5% of the training data is retained as a validation set. Then, after each epoch over the training data, the performance of the resulting model is evaluated on this validation set. If the performance did not improve in the last few epochs, the weights of the model that resulted in the best score on the validation set are loaded, and this model is then finally returned. Since we optimize on fewer training data instances during cross-validation - in comparison to the entire training dataset for the final prediction of the shared task test dataset - it is crucial to automate this process by adjusting the number of training epochs depending on a validation set performance. In our experiments, early stopping was usually executed after 7 to 15 epochs on the training dataset.

2.2 Model Features

In this section we discuss the integration of features from our new POW lexicon (Section 4) into the neural network. We expected the lexicon to provide additional guiding features for offensive language detection, and we experimented with different ways of considering those features as input for the neural network. Computing additional features on the tweet as well as on the word level proved to be beneficial. Furthermore, in our overall network architecture, features based on the lexicon led to performance improvements when including them directly in the text-based sub-network, as well as in the parallel metadata sub-network. We implemented this as follows:

Tweet-level features: For each tweet, we check for each word from the lexicon if it is contained in the tweet (untokenized string). If we find a match, we add 10 feature values to the tweet: one for the word's manually annotated intensity score (0-4) in the lexicon, and nine more for each fine-grained category in the lexicon (0 or 1). If multiple words from the lexicon are matched in a tweet, the values are summed up. For example, a tweet with features [6, 0, 1, 2, 0, 0, 0, 0, 0] might contain 2 words from the lexicon, where each of these has an intensity score of 3, both of them have the RIDICULE label (forth position in the vector), and one has the DEHUMANIZATION label (third position in the vector). Note that it could also be beneficial to only use the score of the matched word with the highest values (instead of the sum); a configuration which we did not test. Since the task

¹Text length, number of proper names, hashtags, etc.

of offensive language detection is formulated as a classification task where we need to identify text that “contains” offensive language, one highly offensive word should be enough to lead to a high prediction score. In our model, we include these 10 features for each tweet as additional metadata features in the parallel sub-network, which then considers a total of 37 features.

Word-level features: In a similar way to the tweet-level features we calculate additional features on word-level. We also apply the above-mentioned procedure on words in a loop on the tokenized tweet, thus resulting in 10 additional features for each word in a tweet. We add these features into the text encoder by stacking them directly on top of the word embeddings. Subsequently, these augmented word embeddings (which are hybrid features containing the distributional semantics of words as well as direct lexicon-based features) are fed into the CNN. The special property of this method is that any generic pre-trained word embeddings can be used, while a task-specific augmentation is included.

Integration into the CNN: Figure 1 displays the architecture of the overall network. The additional tweet-level features are included in the input layer for metadata features on the top right (see `Input_Meta` with dimension 37 for the in total 27 + 10 features). The additional word-level features are given as input in the layer `Input_POW-meta` next to the text input layer (see on top in the middle; last dimension 10 for the 10 features for each word). The 10 dimensions of the additional word-level features are added to the (here: 200) dimensions of the word embedding layer, resulting in augmented word embeddings (see the output of the `Concatenate` layer, last dimension 210).

2.3 Hyperparametrization & Test Results

In this section we give explanations for our choice of parameters and the structure of our final neural network architecture, which was optimized in a 3-fold cross-validation on the GermEval 2019 training dataset. The given numbers are averages over the 3 folds of macro-average F1-scores for offensive language detection (i.e., **Subtask 1**, binary classification). The performance of the basic CNN model without any additional metadata features in the configuration of Schäfer (2018) was 71.98%. Including the basic 27 metadata features in an additional sub-network then resulted in 72.84%. To

measure the effect of using our POW lexicon features, we evaluated different configurations:

- CCN (augmented embeddings) + meta (27 basic features): 73.56%,
- CCN (basic embeddings) + meta (27 basic features + 10 POW): 75.17%,
- CCN (augmented embeddings) + meta (27 basic features + 10 POW): 75.46%.

We can see that both additions seem to be beneficial when activating individual feature categories, while the best results are achieved when we include the lexicon features both on word level as well as on tweet level. A clear improvement can be observed when adding the features on the tweet level, perhaps because the separate sub-network has the capability of learning to identify different types of offensive language than the text encoder sub-network, which could possibly be captured by the lexicon.

We mostly followed the text preprocessing steps described in Schäfer (2018), but we had to make changes to the normalization technique as we ran into a considerable amount of out-of-vocabulary words with the GermEval 2019 dataset. It is important when using word embeddings to have a low number of unknown words, as all such words get assigned a dummy embedding and are basically ignored. We defined our vocabulary as words from the training dataset with a frequency of occurrence greater than 1. This method can be problematic when applied to social media data, due to the high amount of spelling errors, variants or neologisms. Thus, an extensive normalization technique is required.

We attempted to use compound splitting, rule-based/statistical lemmatization, and using the further training data to enrich the vocabulary. However, the best performance was achieved by using a fallback based on *Levenshtein*-similarity for unknown words. When a word is not in the vocabulary, we select the word with the most similar string (i.e., lowest *Levenshtein* distance) instead. This approach is questionable, and in a real-world application we would not suggest it, since it basically guesses unknown words. Nevertheless, it seemed to work here, as it leads to a 1% performance improvement in the given dataset. A proper solution would be to use more training data, which did not work in our evaluation, probably because the training and test sets are too similar. For a

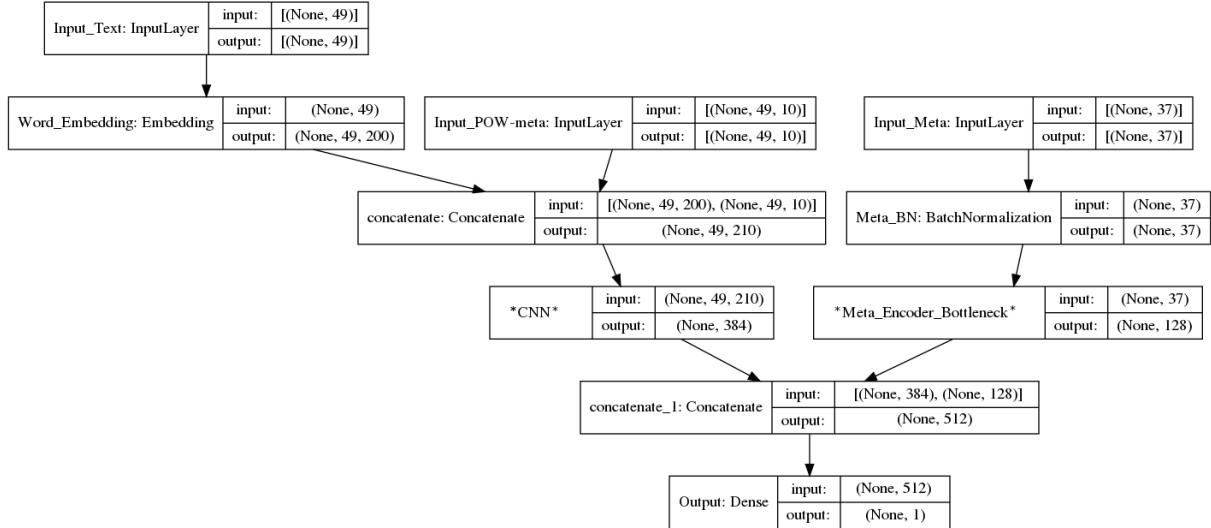


Figure 1: Neural network model structure for binary classification. The blocks correspond to layers, and those marked with * consist of sub-networks (for the exact structures refer to Schäfer and Burtenshaw (2019) for the CNN and to Schäfer (2018) for the Meta_Encoder). Each block contains the name and type of the layer with the dimensions of its input and output. The first value of the dimensions corresponds to the batch size and is given as None in the figure (64 in our experiments).

real-world application it might also be advisable to evaluate on multiple datasets from different sources when developing a system.

For hyperparametrization in general, we found that setting the number of filters of each convolutional layer to 8 leads to the best results. In our final model, we mainly optimized the values for the class weights specifically for each classification task. First, we automatically calculated class weights based on the distribution of the different labels in the training dataset and then optimized these using a smoothing factor.

Class weights: To predict binary labels for offensive language in **Subtask 1**, we used full weights with the values for OTHER: 1.24, and OFFENSE: 2.05. With 3-fold cross-validation on the training dataset, this leads to a macro-average F1-score of 76.37%. For **Subtask 2** (fine grained classification) we found a smoothing factor of 5 applied to the class weights to be optimal. This leads to the weights for OTHER: 1.10, ABUSE: 2.37, INSULT: 2.08, and PROFANITY: 6.06. This means that we try to boost the scores for the infrequent classes, but not as much as their inverse frequency would suggest, since having too many different classes might be detrimental to the overall performance. We achieve a macro-average F1-score of 58.06% using 3-fold cross-validation. For **Subtask 3**, we found it most optimal to use a

smoothing factor 0.5, increasing the values suggested by the inverse frequency of the labels in the dataset, which might be justified by the highly skewed distribution of the two labels. The resulting class weights are EXPLICIT: 1.30, and IMPLICIT: 14.12. With 3-fold cross-validation, this leads to a macro-average F1-score of 64.79%.

Final neural network: Our final neural network (variant for **Subtask 1**) is shown in Figure 1. The network takes three inputs: Input_Text is the tokenized/normalized tweet, Input_POW-meta are the additional word-level features from our POW lexicon, and Input_Meta are the tweet-level metadata features (37 in total: 27 basic + 10 from the lexicon).

To explain the sequence input of the tokenized data into the network, it should be noted that these sequences are all set to a fixed length (49 here, see the 2nd dimension size of the first 2 input layers in the figure). We calculated a suited maximum sequence length of the normalized input tweets to be 49 based on the given training dataset. This value is set so that a maximum of 5% of the tweets has to be cut off, i.e., 95% of the tweets have less than or equal to 49 words after our normalization (such shorter sequences are (pre-)padded). The input sequence (tokenized and normalized tweet) is then transformed into numerical values using the augmented word embeddings (as described above)

and further encoded using a CNN with 6 parallel branches. The (flat) output of the text encoder (see output of the layer CNN; 384 dimensions) is then concatenated with the output of the metadata sub-network (see output of the Meta_Encoder; 128 dimensions) resulting in 512 encoded feature values, on which the final layer produces a value that we can interpret as a binary prediction. We understand the decision-making process of this neural network to be based on two main criteria:

- n-grams of words that hint at the use of offensive language, which the CNN is constructed to identify,
- predefined metadata features based on rules and lexicons (metadata sub-network).

3 HAU3: Random Forest

This model was only trained on this year’s training data, where each message was mapped to a vector of (lowercase) character trigrams features, e.g., *scheiß* = {sch, che, hei, eiß}, and word unigram features. All features are binary (i.e., weight 1 if present in message, weight 0 if not). We used the Random Forest algorithm with the following hyperparameters: a 100 trees, each with a random subset of no more than 750 features, and a minimum leaf node size of 3. The algorithm was written from scratch; an additional incentive was to test its performance in a real-world task and check whether it is eligible for inclusion in our new open source NLP & ML toolkit for the Python programming language, Grasp.py.² This is a smaller, faster and easier-to-use version of our Pattern toolkit (De Smedt and Daelemans, 2012).

In general, the performance of the model is unremarkable: an average F1-score of 69.75% for **Subtask 1**, with a poor recall for OFFENSE tweets (43.71%) and a good recall for OTHER tweets (90.34%). To freshen up, recall and precision can be understood intuitively as follows: suppose we create a dashboard of today’s offensive messages on Twitter. The dashboard shows a 100 messages, which are all offensive, so precision (cf. quality) is 100%. If the list also shows irrelevant messages (false positives) then precision will be lower. Now suppose that in reality there were a 1,000 offensive messages today. This means that the dashboard missed out on 900 (false negatives) and has a low recall (cf. quantity) of only 10%.

²<https://github.com/textgain/grasp>

In this light, in our approach a lot of offensive messages slip through undetected, but few non-offensive messages are misclassified. Arguably, this could still make the classifier useful in real-life, since, from a user experience standpoint, under-blocking (more offensive content slips through) is better than over-blocking (more users are falsely accused of being offensive). A human moderator equipped with a dashboard built on top of our classifier would see about half of the daily offensive content, in a list where he or she would have to ignore about 1/3 of irrelevant results (68.06% precision).

4 HAU2: POW lexicon

This system is based on a lexicon of 2,850 German words that express profanity and offense, with manually-annotated intensity scores. The Profanity & Offensive Word list (POW) originates from our work in last year’s GermEval (see De Smedt and Jaki, 2018), where we used 50 handpicked, high-precision “seed” words to automatically extract 1,250 similar words from the German Twitter Embeddings (Ruppenhofer, 2018). During the past year, the list was further expanded and annotated for intensity (0-4, e.g., *radikal* ‘radical’ = 1, *schwein* ‘pig’ = 3, *abschaum* ‘scum’ = 4) by four annotators at the University of Hildesheim. Each entry in the list also has an English translation, extracted from Google Translate and manually reviewed, and up to 9 fine-grained tags such as PROFANITY, DEHUMANIZATION, RIDICULE, SEXISM, RACISM and/or EXTREMISM, added by the annotators. The set of tags is an open and growing collection. The current tags were chosen intuitively based on prior studies in online German right-wing extremism (Jaki and De Smedt, submitted), misogyny (Jaki et al., 2019), and jihadism (De Smedt et al., 2018). The advantage of such a richly-annotated resource is that offensive words can easily be highlighted (e.g., in a dashboard), making it a useful and explainable support tool for real-world applications where human moderators have the final say.

The classification algorithm is a rule-based script that takes a given message as input, scans which of its words are also in the lexicon, and then returns whether or not it is OFFENSE as output, based on a weighting scheme tweaked by trial-and-error for each variable (i.e., intensity scores + tags). As it turns out, this heuristic approach achieves is not

outperformed a lot by the Random Forest classifier: an average F1-score of 68.13% for coarse-grained **Subtask 1**. Its main weakness is a low recall for OFFENSE (37.11%), which in theory could be overcome by expanding the lexicon with more entries, which is a cost-effective solution when student assistants are available. This is a task that we have planned for future work. For example, our Dutch counterpart lexicon currently has 10,000 entries. Again, moderators of such tools will miss out on half of the offensive content, but it is often more desirable to review 1,000 messages of which 2/3 are really offensive than to wade through 10,000 without any prior warning system.

Interestingly, this approach has a better precision (cf. quality) in the fine-grained task for predicting INSULT (48.28%) than our best CNN (35.56%), presumably because it can depend on handcrafted insights from the PROFANITY + RIDICULE tags. This begs the question what other tags can be useful for future work, with THREAT and ILLEGAL on top of our list.

In related research on political debate (Jaki et al. 2019), we have also used the lexicon to detect offensive content on the Facebook pages of the major political parties in the German federal elections 2017 and their leading candidates. Employing the POW list, we showed that the female candidate’s pages displayed a higher proportion of offensive content, for example.

5 Discussion

In this section, we sum up the HAU results for the different subtasks and discuss their implications.

In GermEval 2018, we have focused primarily on the coarse-grained classification task, with first attempts in the fine-grained classification task (Schäfer, 2018). This year, **Subtask 1** (binary classification) was tackled by employing the three models described in Sections 2 to 4, and **Subtask 2** (fine-grained classification) with the first and last models (CNN and lexicon). We assumed that the most difficult task to solve computationally might be **Subtask 3** (implicit vs. explicit offense) as understanding implicit offense often involves infernal processes that involve a high amount of contextual knowledge. As a consequence, implicit offense is often hard to pin down on the text level.

As the best results were yielded for **Subtask 1** in GermEval 2018 (Wiegand et al., 2018), it is little surprising that the models achieved our best

results for the binary classification task in 2019: a macro-average F1 score of 70.46% for the CNN, 69.75% for the Random Forest, and 68.13% for the POW lexicon. Surprisingly however, the overall performance in **Subtask 3** was higher than in **Subtask 2**, which means that the identification of different types of offense turned out more difficult to solve (macro-average F1 score of 45.34% for the CNN and 40.8% for the POW lexicon) than the identification of implicit offense (macro-average F1 score of 69.3% for the CNN).

The general conclusions we can draw from our submission are the following: Firstly, it is very difficult to outperform CNNs in the detection of offensive language, especially if they have been extensively enriched by a multitude of additional features. Secondly, although even a very substantial lexicon will not outperform more up-to-date approaches, it can still achieve surprisingly good results, in so far as word lists do not necessarily entirely fall behind in comparison to CNNs and Random Forests. This is particularly important to know because there are many real-life scenarios where lexicon-based approaches could be employed as a simple, but useful aid in the detection of offensive speech, with the advantage of bringing a maximum of transparency to the identification process. Thirdly, the integration of lexicons like POW into CNNs can help to trigger better overall results. This shows that we should not exclusively rely on statistical approaches for solving the problem in future work, but recognize the value of thorough qualitative preparation work (such as the result of the annotation process).

References

- Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. *Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter*. In Proceedings of the 13th International Workshop on Semantic Evaluation, pp. 54-63.
- Tom De Smedt and Walter Daelemans. 2012. *Pattern for Python*. JMLR, 13:20632067.
- Tom De Smedt, Guy De Pauw, and Pieter Van Ostaeyen. 2018. *Automatic detection of online jihadist hate speech*. CLIPS CTRS, 7:130.
- Tom De Smedt and Sylvia Jaki. 2018. *Challenges of Automatically Detecting Offensive Language Online: Participation Paper for the Germeval Shared Task*

2018 (*HaUA*). In Proceedings of GermEval 2018, 2732.

Europol. 2019. *European Union Terrorism Situation and Trend Report*. European Union Agency for Law Enforcement Cooperation. https://w019_final.pdfww.europol.europa.eu/sites/default/files/documents/tesat_2.

Sylvia Jaki and Tom De Smedt. 2018, submitted. *Right-wing German hate speech on Twitter: analysis and automatic detection*.

Sylvia Jaki, Tom De Smedt, Maja Gwóźdż, Rudresh Panchal, Alexander Rossa, and Guy De Pauw. 2019. *Online hatred of women in the Incels.me forum: Linguistic analysis and automatic detection*. *Journal of Language Aggression and Conflict*. <http://doi.org/10.1075/jlac.00026.jak>.

Sylvia Jaki, Wolf Schünemann, Tom De Smedt, and Stefan Steiger. 2019. *Who is polluting the debate?* Unpublished conference paper.

Josef Ruppenhofer. 2018. *German Twitter Embeddings*. http://www.cl.uni-heidelberg.de/english/research/downloads/resource_pages/GermanTwitterEmbeddings/GermanTwitterEmbeddings_data.shtml.

Johannes Schäfer. 2018. *HIIwiStJS at GermEval-2018. Integrating Linguistic Features in an Neural Network for the Identification of Offensive Language in Microposts*. In Proceedings of GermEval 2018, 104112.

Johannes Schäfer and Ben Burtenshaw. 2019. *Offence in Dialogues: A Corpus-Based Study*. In Proceedings of Recent Advances in Natural Language Processing (RANLP 2019), pages 1085-1093, Varna, Bulgaria, Sep 2-4 2019.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. *Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language*. In Proceedings of GermEval 2018, 110.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. *Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval)*. arXiv preprint arXiv:1903.08983.

UPB at GermEval-2019 Task 2: BERT-Based Offensive Language Classification of German Tweets

Andrei Paraschiv

Computer Science Department
University Politehnica of
Bucharest, Romania

andrei.paraschiv74@stud.
.acs.upb.ro

Dumitru-Clementin Cercel

Computer Science Department
University Politehnica of
Bucharest, Romania

clementin.cercel@gmail.
.com

Abstract

In this paper, we describe our participation to GermEval-2019 Task 2, which requires identifying and classifying offensive content in German tweets. For all three challenging subtasks, i.e. i) Subtask 1 – a binary classification between Offensive and Non-Offensive tweets, ii) Subtask 2 – a fine-grained classification into three different categories: Profanity, Insult, Abuse and iii) Subtask 3 – detecting whether the tweets contain Explicit or Implicit Offensive language, we used the Bidirectional Encoder Representations from Transformers (BERT) model with a pre-training phase based on German Wikipedia and German Twitter corpora and then performed fine-tuning on the competition dataset. Thus, our approach focuses on how to pre-train, fine-tune and deploy a BERT model to classify German tweets. Our best submission achieves on test data 76.95% average F1-score on Subtask 1, 53.59% on Subtask 2 and 70.84% on Subtask 3.

1 Introduction

Online social networks today are more popular than ever. However, the freedom of communication leads sometimes to abusive and undesired behavior. For example, hate speech, racism, abusive language, doxing or offensive speech has become a real problem for all major online social networks. Due to its short messages and very interactive nature, this behavior is mostly present in Twitter. The huge amount of user-generated content renders a manual review impossible. Bound by the law¹ to remove hate speech from their websites, online media

companies have invested a lot of effort and resources to detect and classify hate speech and abusive language automatically.

The task of abusive and offensive language identification has been recently addressed in several papers and competitions (Kumar et al., 2018) (Zampieri et al., 2019b), with a large focus on English language. The GermEval campaign (Wiegand et al., 2018) tries to overcome this shortage and proposed at the 2019 edition a second shared task on the identification of offensive language in German tweets.

Waseem et al. (2017) identified the dimensionality for the typology of abusive language - the uttering can target a particular person, or it can be directed at a generalized group, for instance, an ethnic minority, immigrants, sexual minority. In the GermEval-2019 Task 2 training set, we can find a higher number of directed abusive statements like “Das Weib hat wirklich einen Vogel”, and some generalized abuse (e.g., “Der Islam hat den Bundesgerichtshof gekauft”). This skewed distribution is consistent with the distribution in the dataset created by (Zampieri et al., 2019a) for English tweets. The other dimension proposed by Waseem et al. is the extent to which the hate speech is explicit or implicit and it is directly addressed by our Subtask 3.

In our research, we deployed a deep learning system based on Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018), a general language model that is by default pre-trained on two corpora, i.e., English Wikipedia and BooksCorpus (Zhu et al., 2015), using a “masked language model” and “next sentence prediction”. In contrast to classical word embedding models like GloVe (Pennington et al., 2014) or Word2Vec (Mikolov et al., 2013), BERT uses a limited vocabulary

¹https://www.bmjv.de/SharedDocs/Gesetzgebungsverfahren/Dokumente/NetzDG_engl.pdf

(around 30,000 words, compared to 400,000 words used by GloVe), since it relies not only on word embeddings but also on segment and positional embeddings. Our paper assesses the performance of a slightly modified BERT model, pre-trained from scratch on the German Wikipedia followed by German Twitter data.

The remainder of this work is organized as follows. The next section briefly shows an overview of the current methods employed for detecting offensive language. Section 3 describes the GermEval-2019 Task 2 dataset and our runs for this task. Section 4 illustrates the performances of each run. Section 5 outlines the conclusions that can be drawn from our work and possible future improvements.

2 Related Work

In recent years, the issues of toxic comments and abusive language have come to the forefront of text classification research. Generally, most classification studies have focused on three main topics:

- Identifying offensive texts (binary classification);
- Distinguishing between explicit and implicit offensive language;
- A fine-grained classification of offensive texts.

One major problem for researchers is the difficulty to clearly define hate speech, and also the lack of large labeled datasets, and thus the lack of consensus among the annotators (Waseem, 2016). Moreover, in most large corpora, the percent of offensive speech is very low and labeling enough positive samples takes a lot of tedious work. Gilbert et al. (2018) narrows the search down by choosing a white supremacist forum to extract the samples and then to manually annotate them. Their work not only provides an insightful annotation procedure but also shows that, for an accurate labeling, a sentence needs extra context, for example, the whole conversation or the forum thread title.

For German language, Ross et al. (2017) showed that reliability in the annotation work is relatively low and more guidelines and clear definitions can bring improvements. Notably, Köffer et al. (2018) has shown that methods developed for hate speech detection in English language can be successfully applied to similar tasks in German language. However, due to German language characteristics, the process can

be more complex, and results might achieve lower scores than their English counterparts.

Burnap and Williams (2015) used a feature-based classification employing various machine learning algorithms - Bayesian Logistic Regression, Random Forest Decision Trees, Support Vector Machines and an ensemble of all three models. They used different feature sets, such as n-grams, hateful terms and typed dependencies.

Recently, deep learning models have been widely applied to handle natural language processing tasks. For example, Gao and Huang (2017) proposed the utilization of context information by employing BiLSTM (Bidirectional Long-Short Term Memory Networks) with attention layer (Bahdanau et al., 2014) for hate speech detection. Founta et al. (2018) developed a deep neural network architecture that also takes various tweet metadata into account (e.g., number of followers, number of retweets, etc.) besides the content of the tweets. Schäfer (2018) is building upon this architecture and proposed a classification model aimed at German texts.

Wu et al. (2019) used the BERT model to detect and classify offensive language in English tweets. They used the base, uncased version with 768-dimensional embeddings and obtained good results in the binary classification task.

3 Methodology

3.1 Data analysis

For both Subtasks 1 and 2 (i.e., binary and fine-grained classification respectively), the training dataset supplied for the GermEval-2019 competition Task 2 consists of 3,995 annotated German tweets. Additionally, for Subtask 3 (i.e., explicit or implicit offensive language classification) the training dataset contains 1,958 annotated German tweets.

Annotations for the Subtask 1 classification were OFFENSE for the positive class and OTHER for the negative class. For Subtask 2, the positive class was split into three categories, i.e., PROFANITY, INSULT, ABUSE. Subtask 3 annotations were EXPLICIT and IMPLICIT, since all tweets in this case were marked as OFFENSE.

3.2 Additional training set

To increase the training data size, we also used the annotated data from the previous GermEval-2018 edition (Wiegand et al., 2018), including

8,541 German tweets. This additional dataset was only suitable for Subtasks 1 and 2, since in this dataset there were no implicit/explicit labels. Table 1 shows the final distribution of classes in our training set for Subtasks 1 and 2 and Table 2 presents the distribution for Subtask 3.

Class	Tweets	%
Other	8,359	66.70%
Profanity	271	2.10%
Insult	1,601	12.80%
Abuse	2,305	18.40%
Offensive	4,177	33.30%
Total	12,536	100.00%

Table 1: The final distribution of the tweets in the training dataset for both Subtasks 1 and 2

Class	Tweets	%
Explicit	1,699	86.80%
Implicit	259	13.20%
Total	1,958	100.00%

Table 2: The distribution of the tweets in the training dataset for Subtask 3

3.3 Data preprocessing

All tweets were pre-processed before the classification step. Some basic replacements were performed:

- Emojis encoded in strings like <U+0001F44D> were converted to their unicode representation;
- Emoji characters were spelled out into words like <thumbs_up> or <rolling_on_the_floor_laughing>;
- Usernames, weblinks and newline markers were converted to the standard tokens <user>, <url> and <nl>;
- Numbers, dates and timestamps were converted to standard tokens <number> and <time> using the Ekphrasis text processing tool (Baziotis et al., 2017).

Further, we tried to split each hashtag into atomic words, for instance, hashtags like #EheFürAlle into “Ehe für Alle”. Since not all hashtags are camel-cased, we tried to split all non-camel-cased hashtags by using unigrams and bigrams (Baroni et al., 2009).

We checked for spelling errors and unresolved hashtags using the German GloVe vocabulary² as reference. Abbreviations,

misspellings and spaced out words were manually replaced, for instance, “E N D L I C H” to Endlich, #noAfD to “no AfD”, “innenminist” to “Innenminister”, “soooooooo” to “so”, and “schonlängerhierlebende” to “schon länger hier lebende”.

Due to the fact that German words can change their meaning for different capitalization, we tried to preserve or correct the upper/lower case of words.

3.4 Model description

We used the BERT-Base, cased, model pre-trained from scratch using German Wikipedia, OpenLegalData corpus and news articles by deepset.ai³.

Model BERT (Devlin et al., 2018) is a bidirectional model and consists of 12 transformer blocks, 12 attention heads and 110M parameters. There are two pre-training phases for BERT: “masked language modeling” and “next sentence prediction”. For masked language modeling, the model predicts the probabilities for a percentage of random “masked” words from a sentence. The next sentence prediction phase trains the language model to predict if one sentence might follow another sentence.

Pre-training Since tweets have a specificity not captured by Wikipedia or news articles, we pre-trained the BERT model on the 6.2M tweets, consisting of three corpora of German tweets as follows:

- A corpus of 1,212,220 tweets collected by Kratzke (2017) in the context of German Federal Elections of 2017;
- A corpus of 5,964,889 tweets collected by Kratzke (2019) in the months of April and May 2019 around the European Elections 2019;
- A collection of 70,745 tweets of well-known trolling or aggressive Twitter accounts, namely SiffTwitter⁴, that were collected by us using Tweepy⁵.

For the purposes of this step, we pre-processed the above-mentioned Twitter corpora similar to the training data and experimented with the pre-training hyperparameters. The optimal results were achieved for 150,000 training steps with 10,000 warmup steps and a learning rate of 0.0001.

³ <https://deepset.ai/german-bert>

⁴ <https://medium.com/@trolltwitter/sifftwitter-infos-uber-die-schlimmste-hasscommunity-im-netz-dcf1f943c0227>

⁵ <https://www.tweepy.org/>

² <https://deepset.ai/german-word-embeddings>

Unlike models based on GloVe or Word2Vec, BERT uses by default a WordPiece tokenization (Schuster and Nakajima, 2012), that helps to reduce the vocabulary file to ~31,000 words. Additionally, we extended the BERT vocabulary file with 181 frequent out-of-vocabulary words (e.g., “Rechtspopulismus”, “Migrationspakt”, “Ibizagate”, etc.) from our training data and pre-training corpora. Using WordPiece, the model handles out-of-vocabulary words by breaking them in subwords. For example, words like “Versprechungen” that are not in the vocabulary are tokenized into “Versp”, “##rech”, “##ungen”, or “einlösen” into “ein”, “##lösens”.

Classification For the classification step, we modified the original BERT model. For Subtasks

1 and 3, we removed the last nine layers from the model. Then, we added a LSTM layer and its output is fed into a fully connected layer with a two-dimensional output vector. In contrast, for Task 2, we removed the last six layers from the model and added a fully connected layer with a three-dimensional output vector, since we predicted only the labels for the entries that were detected as offensive in Task 1.

Fine-tuning The model-training stage was performed with all 12,536 tweets for Subtasks 1 and 2 respectively and with 1,958 tweets for Subtask 3.

Submissions We submitted three runs for evaluation on the test data. The first one was based on above mentioned steps. The second run

Model	Accuracy	Precision	Recall	F1
BiLSTM	76.94%	73.49	73.06	73.27
TUWienKBS	75.32%	72.43	74.49	73.44
BERT Run 1	79.38%	76.35	77.55	76.95
BERT Run 2	79.64%	76.6	77.12	76.86
BERT Run 3	79.38%	76.35	77.55	76.95
BERT no pre-train	78.62%	75.51	76.64	76.07

Table 3: Performance comparison of various models on test data for Subtask 1. Precision, Recall and F1-measure are average values over the two classes (OTHER, OFFENSE). The best result is shown in boldface.

Model	Accuracy	Precision	Recall	F1
BiLSTM	67.44%	50.53	39.97	44.64
TUWienKBS	70.47%	53.21	49.42	51.24
BERT Run 1	73.61%	58.53	49.42	53.59
BERT Run 2	71.66%	54.4	50.7	52.48
BERT Run 3	73.57%	55.63	49.02	52.11
BERT no pre-train	70.01%	53.04	47.3	50.01

Table 4: Performance comparison of various models on test data for Subtask 2. Precision, Recall and F1 are average values over the four classes (PROFANITY, INSULT, ABUSE and OTHER). The best result is shown in boldface.

Model	Accuracy	Precision	Recall	F1
BiLSTM	85.59%	42.80	50.00	46.12
BERT Run 1	87.85%	77.44	65.28	70.84
BERT Run 2	86.88%	73.88	63.48	68.29
BERT Run 3	87.20%	74.39	66.77	70.37
BERT no pre-train	86.13%	71.35	66.76	68.98

Table 5: Performance comparison of various models on test data for Subtask 3. Precision, Recall and F1 are average values over the two classes (EXPLICIT and IMPLICIT OFFENSE). The best result is shown in boldface.

had an additional pre-processing step compared to Run 1, i.e. the words that were not in the GloVe vocabulary were split if possible by using unigrams and bigrams (Baroni et al., 2009), in order to tackle the problem of German compound words. Run 3 was similar to that of Run 1, only instead of 2 epochs, we trained the model for 4 epochs.

For baseline comparison, we used the best performing model at GermEval-2018, namely TUWienKBS, proposed by Montani and Schüller (2018) and also a BiLSTM-based model with German GloVe embeddings⁶ of 300 dimensions and a vocabulary of 20,000 words.

4 Experiments

The results for Subtask 1 of the three runs and the baseline models are given in Table 3. As can be seen, the additional preprocessing step did not increase the score for Run 2, since it increased the precision but lowered the recall.

Table 4 shows the results for the fine-grained classification. Thus, the additional training epochs did not improve the results, on the contrary, it seemed to overfit the model.

Finally, as seen in Table 5, the model performed well on the explicit/implicit task and we can see that the additional preprocessing step decreased the score for Run 2.

For Subtask 1, we can see in Figure 1 the learning curve of the average F1-score for one training epoch. After 90% of the training set, the score improvement is less significant and even with 50% of the training data, we can reach good scores.

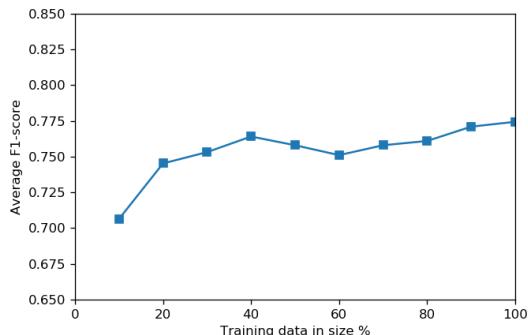


Fig1: Run 1 learning curve of average F1-score

Confusion matrices for the run with the best score, namely BERT Run 1, can be viewed in Tables 6-8 for each subtask. As we can see in Table 6, the model has the tendency to

underpredict offensive content. Tweets as “@morgenpost *Das ist eine Sie? h...*” or “Jetzt daheim. Vielen Dank an den Hurentisch 30, der noch eineinhalb Stunden nach Ladenschluss fröhlich Caipirinas bestellt hat.” prove difficult to predict as OFFENSE.

The model has also difficulty in telling INSULT and ABUSE apart (see Table 7). Tweets like “@SPIEGELONLINE *Merkel eine Schande für Deutschland überall machen wir Deutsche uns zum Narren*” or “Deutschland, die anderen 149 Länder wollen ihre Kriminelle Unterschicht loswerden. @WELT_Politik” were classified as ABUSE rather than INSULT.

Finally, due to the imbalance in the classes for Subtask 3, the model has the tendency to over predict explicit offenses. Tweets like “Infotweet: Es gibt nur 2 Geschlechter #GenderDay” or “Immer wenn ich Deutsche Kinder sehe krieg ich wieder Hoffnung für mein Land” were wrongly classified as EXPLICIT.

True Label	Predicted Label	
	OTHER	OFFENSE
OTHER	1825	236
OFFENSE	330	640

Table 6: The confusion matrix of BERT Run 1 model for Subtask 1

True Label	Predicted Label			
	OTH	PROF	INS	ABU
OTH	1825	6	89	141
PROF	37	19	25	30
INS	171	7	147	134
ABU	122	2	36	240

Table 7: The confusion matrix of BERT Run 1 model for Subtask 2.

True Label	Predicted Label	
	IMPLICIT	EXPLICIT
IMPLICIT	45	89
EXPLICIT	24	772

Table 8: The confusion matrix of BERT Run 1 model for Subtask 3.

⁶ <https://deepset.ai/german-word-embeddings>

5 Discussion and Conclusions

In this study, we used the BERT-Base version in order to classify German tweets into different categories related to offensive language. Our results show that BERT is a powerful model, capable of detecting offensive language accurately. BERT outperforms a BiLSTM model with GloVe word embeddings for all three subtasks, and the TUWienKBS model in both Subtasks 1 and 2. However, a more subtle classification into nuances of offensive language can lead to lower scores. This can also be a result of the highly unbalanced three subcategories, or due to an unclear delimitation between them.

As for the detection of implicit versus explicit offensive language, it achieves a higher score if one takes into account the fact that the error from Subtask 1 will propagate into Subtask 3. As seen in the results, the pre-trained model with political targeted tweets leads to a slight increase in performance for the binary tasks, but a significant increase for the fine-grained classification.

For future work, we noticed that the treatment of emoticons could be significantly improved, since the spelling out of certain emojis does not always improve the detection. Additionally, a larger set of tweets for pre-training would improve the language understanding of the model. Also, we will work to better distinguish between Insult and Abuse languages, which would allow us to improve the results for Subtask 2.

References

- Dzmitry Bahdanau, Kyunghyun Cho and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*
- Marco Baroni, Silvia Bernardini, Adriano Ferraresi and Eros Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. In *Language Resources and Evaluation*, 43(3): 209–226.
- Pete Burnap and Matthew L Williams. 2015. Cyber Hate Speech on Twitter: An application of machine classification and statistical modeling for policy and decision making. In *Policy & Internet* 7(2):223–242.
- Christos Baziotis, Nikos Pelekis and Christos Doulkeridis. “DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis.” SemEval@ACL (2017).
- Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv abs/1810.04805*
- Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2018. A unified deep learning architecture for abuse detection. *CoRR, abs/1802.00385*
- Lei Gao and Ruihong Huang. 2017. Detecting online hate speech using context aware models. *arXiv preprint arXiv:1710.07395*
- Ona de Gibert, Naiara Pérez, Aitor García Pablos and Montse Cuadros. 2018. Hate Speech Dataset from a White Supremacy Forum. In *ALW2: 2nd Workshop on Abusive Language Online*, pages 11–20.
- Sebastian Köffer, Dennis M. Riehle, Steffen Höhenberger and Jörg Becker. 2018. Discussing the Value of Automatic Hate Speech Detection in Online Debates. In *Multikonferenz Wirtschaftsinformatik (MKWI 2018): Data Driven X - Turning Data in Value, Leuphana, Germany*.
- Nane Kratzke. 2017. The #BTW17 Twitter Dataset - Recorded Tweets of the Federal Election Campaigns of 2017 for the 19th German Bundestag [Data set]. Data. Zenodo. <http://doi.org/10.5281/zenodo.835735>
- Nane Kratzke. 2019. Monthly Samples of German Tweets (Version 2019-05) [Data set]. Zenodo. <http://doi.org/10.5281/zenodo.3236750>
- Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC)*, Santa Fe, USA.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*, 26, pages 3111–3119.
- Joaquin Padilla Montani and Peter Schüller. 2018. TUWienKBS at GermEval 2018: German Abusive Tweet Detection. In *Proceedings of GermEval 2018, 14th Conference on Natural Language Processing, KONVENS 2018*.
- Jeffrey Pennington, Richard Socher and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing*, pages 1532–1543.
- Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky and Michael

- Wojatzki. 2016. Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis. *ArXiv abs/1701.08118*
- Johannes Schäfer. 2018. HIIwiStJS at GermEval-2018: Integrating Linguistic Features in a Neural Network for the Identification of Offensive Language in *Microposts Proceedings of the Workshop Germeval 2018 - Shared Task on the Identification of Offensive Language*. Vienna, Austria.
- Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conferenceon, pages 5149–5152. IEEE.
- Zeerak Waseem. 2016. Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142.
- Zeerak Waseem, Thomas Davidson, Dana Warmsley and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. *ALW@ACL arXiv:1705.09899*.
- Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of the GermEval Workshop*. Vienna, Austria, pages 1–10.
- Zhenghao Wu, Hao Zheng, Jianming Wang, Weifeng Su and Jefferson Fong. 2019. BNU-HKBU UIC NLP Team 2 at SemEval-2019 Task 6: Detecting Offensive Language Using BERT model. In *SemEval 2019 at North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 551-555.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.
- Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

hpiDEDIS at GermEval 2019: Offensive Language Identification using a German BERT model

Julian Risch¹

Anke Stoll²

Marc Ziegele³

Ralf Krestel⁴

¹Hasso Plattner Institute, University of Potsdam, julian.risch@hpi.de

^{2,3}Heinrich Heine University Düsseldorf, anke.stoll@hhu.de, ziegele@phil.hhu.de

⁴University of Passau, ralf.krestel@uni-passau.de

Abstract

Pre-training language representations on large text corpora, for example, with BERT, has recently shown to achieve impressive performance at a variety of downstream NLP tasks. So far, applying BERT to offensive language identification for German-language texts failed due to the lack of pre-trained, German-language models. In this paper, we fine-tune a BERT model that was pre-trained on 12 GB of German texts to the task of offensive language identification. This model significantly outperforms our baselines and achieves a macro F1 score of 76% on coarse-grained, 51% on fine-grained, and 73% on implicit/explicit classification. We analyze the strengths and weaknesses of the model and derive promising directions for future work.

1 Offensive Language in Online Media

Social media, micro-blogging, and comparable participatory platforms can offer freely accessible discussion spaces and the possibility of communicative integration of different social and interest groups. For this reason, they represent an important cornerstone of modern democracies. In reality, however, online discussions are often the scene of violence, abuse, and incivility (Coe et al., 2014). Studies have shown that offensive and abusive communication makes participants withdraw from online discussions (Springer et al., 2015). Additionally, offensive language can promote aggressive cognitions and negative emotions (Rösner et al., 2016), and reinforce negative prejudices against social groups (Hsueh et al., 2015). The automated detection of offensive language and related concepts, such as incivility, hate speech, or toxicity could help to counter such effects by supporting moderators in effectively identifying and responding to offensive content in online discussions.

This paper presents an approach of detecting different forms of offensive language including profanity, insult, and abuse, and explicit and implicit offensive language in German-language tweets using BERT (Bidirectional Encoder Representations from Transformers). In the GermEval Shared Task 2 (2019), our best systems achieve macro F1 scores of 76.4% on coarse-grained, 51.2% on fine-grained, and 73.1% on implicit/explicit classification.

2 Related Work

Offensive language identification and related tasks, such as the detection of toxicity and hate speech, have recently gained popularity within the Natural Language Processing (NLP) community. These tasks are particularly challenging from an NLP perspective, since hate speech, toxicity, or offensive language are often not explicitly communicated through the use of unique offensive words. Further, many words are used with different meanings in different contexts. Traditional lexical and bag-of-words (BoW) approaches often struggle in identifying implicit and context-related forms of offensive language. Davidson et al. (2017) found that only five percent of tweets that contain words of the hate speech lexicon Hatebase.org were flagged as hate speech by human annotators. In their study on anti-black racism on Twitter, Kwok and Wang (2013) show that tweets are classified as abusive based on words such as “black” or “white”, which bear no racist undertones of their own.

Deep Learning (DL) methods marked a significant step forward in the detection of several forms of offensive or abusive language. They enable the use of word vectors, e.g., Word2Vec (Mikolov et al., 2013), Glove (Pennington et al., 2014), or ELMo (Peters et al., 2018) instead of bag-of-words representations. Further, DL models such as Long Short Term Memory (LSTM) networks or Convolution Neural Networks (CNN) achieved significantly better results in several NLP tasks than less com-

plex classifiers, such as Support Vector Machines, Logistic Regression or Decision Tree Models. Badjatiya et al. (2017) experimented with multiple DL architectures and text representations to detect hate speech on a dataset of 16,000 annotated English-language tweets. They demonstrated that DL approaches, in sum, outperform models based on character and word n-gram representations.

However, such models need extensive amounts of (manually labeled) training data and are often bound to the training data structure and the specific task they are trained for. As a consequence, it is problematic to apply these models to other tasks, such as detecting the outcome of the model in other languages. In our study, we applied BERT language models (Devlin et al., 2018) to detect different forms of offensive language in German tweets. BERT is not developed to solve a specific problem but constitutes a general language model. That means, BERT does not learn, e.g., what words occur in offensive tweets, but learns how words of a language (e.g. English) are generally organized and combined (Devlin et al., 2018). BERT uses an approach called “masked language model” (MLM), that allows bidirectional learning, meaning learning context both to the right and to the left of words, which previous models were not designed to do. English-language BERT models have been used in other shared tasks, such as SemEval-2019 Task 6: “Identifying and Categorizing Offensive Language in Social Media” (Zampieri et al., 2019). However, to the best of our knowledge there are no publications about German-language BERT models.

3 Dataset and Tasks

We trained our models on a dataset of German-language tweets provided in context of the GermEval Shared Task 2 (2019) on the identification of offensive language. The task consists of three classification subtasks: subtask I is a binary classification whether a tweet contains offensive language or not (coarse-grained); subtask II requires distinguishing between three subcategories of offensive language (fine-grained); and the goal of subtask III is to decide whether offensive tweets are implicit or explicit offensive. Figure 1 shows example tweets from the training data for each category.

3.1 Coarse-Grained Binary Classification

Subtask I is to decide whether a tweet includes some form of offensive language or not. For this

@RoemeltA Du bist jetzt geblockt, denn rassistische Kackscheisse höre ich mir nicht an, ich lese sie nicht und noch viel weniger diskutiere ich darüber. Punkt. *OFFENSIVE*

@MiKeyyy328 schon ok ich verstehe das *OTHER*

(a) Training samples for coarse-grained classification.

@Sternenrot @_schwarzeKatze aber das ist halt einfach kein topf wtf *PROFANITY*

@Dr_Dicht Selber *SCHULD*, wenn Sie hässliche NAPFSÜLZE auch damit aufhören! *INSULT*

@Hallaschka_HH Antisemitismus gehört zur DNA von Luthers Kirche. *ABUSE*

(b) Training samples for fine-grained classification.

@krippmarie Ich kenne noch einige Namen unter den SPDler die ebenfalls zu Grabe getragen müssten-sollten-werden.... *IMPLICIT*

@diMGiulia1 Araber haben schon ekelhafte Fressen....! *EXPLICIT*

(c) Training samples for implicit/explicit classification.

Figure 1: Example tweets and their class labels.

task, 1,282 tweets labeled as *OFFENSIVE* and 2,698 tweets labeled as *OTHER* were used. Figure 1a shows examples of both categories.¹

3.2 Fine-Grained Multi-Class Classification

The goal of subtask II is to detect the subcategories of offensive language, namely *PROFANITY*, *INSULT*, and *ABUSE*. *PROFANITY* is simply the usage of profane words in non-insulting contexts. *INSULT*, unlike profanity, requires an intention to offend an individual or a group. A tweet is labeled as *ABUSE* if it not only insults a person but also includes the stronger form of abusive language. In the dataset for subtask II, 152 tweets are labeled as *PROFANITY*, 624 are labeled as *INSULT* and 506 as *ABUSE*. Figure 1b shows examples of all three categories.

¹Disclaimer: The examples may be considered profane, vulgar, or offensive. They do not reflect the views of the authors and exclusively serve to explain linguistic patterns.

3.3 Implicit/Explicit Classification

Subtask III aims at classifying tweets as either implicit or explicit offensive. 257 tweets are labeled as implicit and 1,664 tweets are labeled as explicit offensive language. Figure 1c shows examples of both categories.

4 Fine-Tuning BERT for German Tweets

In this section, we describe our German-language BERT model and we present our simple, yet effective, ensembling strategy. Further, we detail our submitted runs.

4.1 BERT

The core of our approach is a case-sensitive German-language BERT model. It is pre-trained on 12 GB of raw text from the German-language Wikipedia dump, OpenLegalData dump, and news articles.² The model is of the same size as the English-language “BERT-Base” model (12-layers, 768-hidden, 12-heads) and comprises 110 million parameters. We use the framework FARM³ and make our implementation available online.⁴

Tweets are padded/clipped to a maximum length of 150 tokens each. The average length in the training dataset is 41 words and less than 0.2 percent of the tweets are clipped. For fine-tuning BERT, we use a batch size of 32. Smaller training batches would most likely not contain samples from all classes. We use the Adam optimizer with an initial learning rate of $2e-5$ and warmup the learning rate on 10 percent of the training data — compared to 1 percent of the data in the original BERT paper (Devlin et al., 2018). Other parameters, such as a 10 percent embedding dropout rate, are the same as in the original paper. A weighted cross-entropy loss takes into account the unbalanced class distribution in the training data. For example, regarding the fine-grained classification subtask, the class weights are 1.96 (*ABUSE*), 6.57 (*PROFANITY*), 1.56 (*INSULT*), and 0.37 (*OTHER*). The training runs for one to four epochs, depending on the exact submission described in Section 4.3.

We optionally apply two preprocessing methods. First, we replace all user mentions, such as `@Pe_ter`, with the token *Name*. This normalization helps to reduce the variety of different user

names without losing information about the entity type. Second, for the implicit/explicit classification task, both training and test set provide the true fine-grained class labels for each tweet. We append these class labels as additional text tokens at the end of each tweet to incorporate this information into our model.

BERT uses tokenized parts of words instead of tokenized words. We give two examples of this tokenization.

text: @RobertHabeck Ihr verunglücktes Videostatement hat doch rein gar nichts mit Twitter zu tun. Sie hätten dies ja auch in irgendeine Kamera eines TV-Teams hineinsprechen können.

tokens: [’Name’, ’Ihr’, ’ver’, ’##unglück’, ’##tes’, ’Videos’, ’##tat’, ’##ement’, ’hat’, ’doch’, ’rein’, ’gar’, ’nichts’, ’mit’, ’Twitter’, ’zu’, ’tun’, ’.’, ’Sie’, ’hätten’, ’dies’, ’ja’, ’auch’, ’in’, ’irgend’, ’##eine’, ’Kamera’, ’eines’, ’TV’, ’-’, ’Teams’, ’hinein’, ’##sprechen’, ’können’, ’.’]

Note that the tokenization correctly separates *hineinsprechen* into *hinein* and *sprechen*, *irgendeine* into *irgend* and *eine*, and *verunglücktes* into *ver*, *unglück* and *tes*.

text: @Dr_Dicht Selber SCHULD, wenn Sie hässliche NAPFSÜLZE auch damit aufhören!

tokens: [’Name’, ’Sel’, ’##ber’, ’SC’, ’##H’, ’##U’, ’##L’, ’##D’, ’.’, ’wenn’, ’Sie’, ’hä’, ’##ss’, ’##liche’, ’NA’, ’##P’, ’##FS’, ’##Ü’, ’##L’, ’##Z’, ’##E’, ’auch’, ’damit’, ’auf’, ’##hören’, ’[UNK]’]

Note that the exclamation mark at the end of the tweet is treated as an unknown symbol because the pre-trained language model discards all punctuation marks as a preprocessing step. As a consequence, our classifier does not distinguish question marks and exclamation marks and treats both as unknown symbols. Further, the tokenization does not correctly deal with words written with all capitals, such as *SCHULD*. In general, we find that uppercase letters followed by another uppercase letter are interpreted as a single token. Exceptions are abbreviations that the tokenizer learned and that are

²<https://deepset.ai/german-bert>

³<https://github.com/deepset-ai/FARM>

⁴<https://hpi.de/naumann/projects/repeatability/text-mining.html>

written with all capitals, such as *SC* for *Sportclub*. We assume that this abbreviation is learned because it frequently occurs in the pre-training data for the German BERT model, such as news articles.

4.2 Ensembling Strategy

Fine-tuning BERT is about tailoring the language model to a particular downstream task, such as sequence classification or question answering. An additional output layer, called prediction head, is appended to the model and trained on labeled data. The predictions of our BERT model vary when using different random weight initializations for the prediction head. Therefore, we create an ensemble of the predictions of multiple models. To this end, we use soft majority voting:

$$\hat{y} = \operatorname{argmax}_j \sum_{i=1}^n p_{i,j}$$

where $p_{i,j}$ is the probability for class label j predicted by the i -th classifier (out of n classifiers). We ensemble five runs for the binary classification tasks (coarse-grained and implicit/explicit) and ten runs for the multi-class classification task (fine-grained).

4.3 Submitted Runs

For each of the three tasks, we submitted three runs as described here. For the coarse-grained classification:

- hpiDEDIS_coarse_1 one training epoch
- hpiDEDIS_coarse_2 two training epochs
- hpiDEDIS_coarse_3 four training epochs

For the fine-grained classification:

- hpiDEDIS_fine_1 one training epoch
- hpiDEDIS_fine_2 two training epochs
- hpiDEDIS_fine_3 four training epochs

For the implicit/explicit classification:

- hpiDEDIS_implicit_1
two training epochs, normalized user names
- hpiDEDIS_implicit_2
four training epochs
- hpiDEDIS_implicit_3
four training epochs, normalized user names

Table 1: Macro-average F1 scores on test data.

Run	Ensemble			Single Model		
	Coarse	Fine	Imp.	Coarse	Fine	Imp.
1	75.3	42.0	70.8	74.4	41.0	70.5
2	76.4	47.1	73.1	75.5	45.7	72.1
3	76.1	51.2	73.1	75.4	49.1	72.4
Base	-	-	-	50.0	34.3	62.0

5 Results

We present the results for the identification of offensive language using BERT German-language models. We further describe our baseline approach to compare the results for each of the subtasks. Table 1 lists the macro-average F1 scores of the BERT models and the baseline approach on the test dataset for each of the three tasks. It further compares the results of the ensemble models to the single models. It can be seen that the ensemble models always outperform the corresponding single models. For example, for the fine-grained classification task, the best single model achieves a score of 49.1 compared to a score of 51.2 with the best ensemble model (4 percent improvement). All BERT based models clearly outperform the baseline models by up to 26 percentage points.

5.1 Baseline Approach

As a baseline approach, we applied a Logistic Regression model on a tf-idf weighted bag-of-words (BoW) feature representation of unigrams, bigrams and trigrams (Risch et al., 2018). For preprocessing, the tweets have been tokenized applying the NLTK TweetTokenizer⁵, which we also used to remove Twitter handles and to normalize word tokens such as *duuuuummm* and *duuuuuuummmmm* to a common word token *duuumm*. We further removed stopwords, using the Stopwords ISO⁶ word list for German. For subtask III (implicit/explicit offensive language classification) we added the information if a tweet was abusive, profane, or insulting as a feature, which shows whether one of these forms is more likely to be expressed implicitly or explicitly.

5.2 Coarse-Grained

The coarse-grained classification task is the simplest of the three subtasks and our ensemble BERT

⁵<https://www.nltk.org/api/nltk.tokenize.html>

⁶<https://github.com/stopwords-iso>

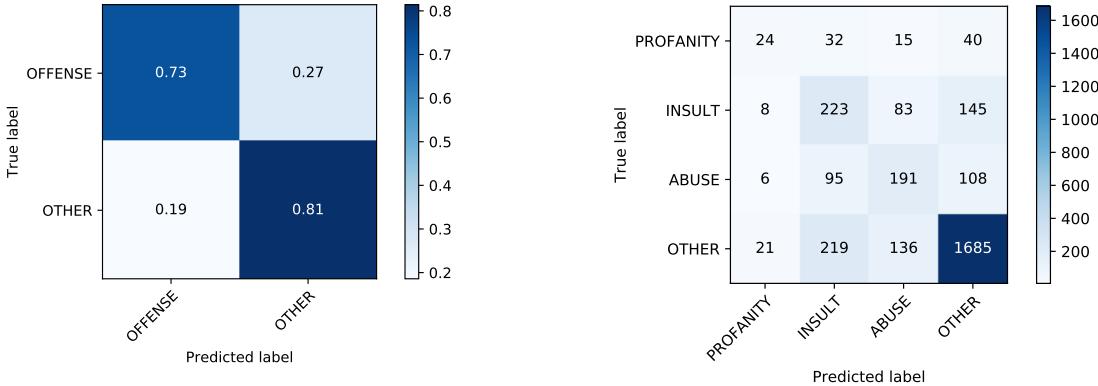


Figure 2: Normalized confusion matrix for the coarse-grained classification subtask.

model achieves a macro-average F1 score of 76.40 percent after two training epochs. Training for one epoch (75.26 percent) or four epochs (76.06 percent) yields similar performances. Figure 2 shows the normalized confusion matrix for the task. The row-based normalization discards the influence of the imbalanced class distribution so that all classes are considered to be equally important. It shows that the model is more reliable when identifying the *OTHER* class than the *OFFENSE* class. 81 percent of the non-offensive and 73 percent of the offensive tweets have been retrieved by the model. The baseline model, on the other hand, retrieved 60 percent of the offensive and 46 percent of the non-offensive tweets.

5.3 Fine-Grained

Figure 3 shows two confusion matrices for the fine-grained classification task. The upper subfigure uses the absolute number of samples, while the lower subfigure uses normalized numbers. The confusion matrix reveals that *OTHER* is identified most reliably by far. *INSULT* and *ABUSE* are equally well identified (recall 0.49 and 0.48). However, in percentage terms, *INSULT* is more frequently mistakenly confused with *OTHER* than with *ABUSE*. *PROFANITY* is most challenging to identify and is most frequently confused with *OTHER* and least frequently confused with *ABUSE*. This confusion matches the similarity of the classes. The baseline model struggles with the distinction of the four subcategories of offensive language and receives only 16 percent recall for *PROFANITY*, which was highly underrepresented in the training data. Recall for *INSULT* is 0.28, for *ABUSE* 0.26 and for *OTHER* 0.67.

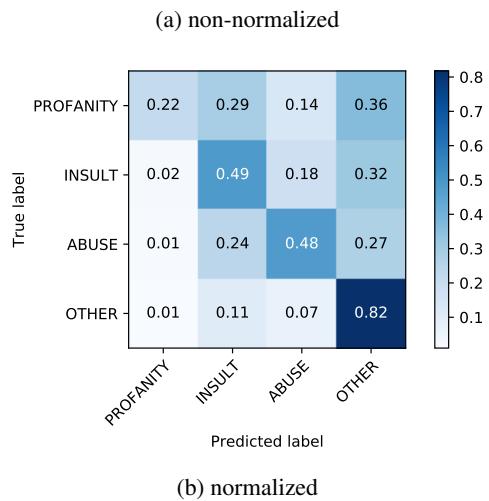


Figure 3: Confusion matrices for the fine-grained classification subtask.

5.4 Implicit/Explicit

Figure 4 shows the normalized confusion matrix for subtask III on implicit and explicit offensive language classification. Implicit offensive language is way harder to identify than explicit offensive language. Most of the explicit offensive tweets are identified by the model (recall 0.92), but only about half of the implicit offensive tweets. The model makes most of the mistakes by misclassifying implicit tweets as explicit (recall 0.54). The baseline BoW-classifier for this task makes similar mistakes but performs worse on the detection of implicit offensive language. Its recall for explicit is 0.91 and 0.31 for implicit offensive language. Again, the implicit offensive language category was highly underrepresented, which probably made the model choose the explicit class when in doubt.

6 Conclusions and Future Work

We studied the problem of offensive language identification in the context of the GermEval task 2019.

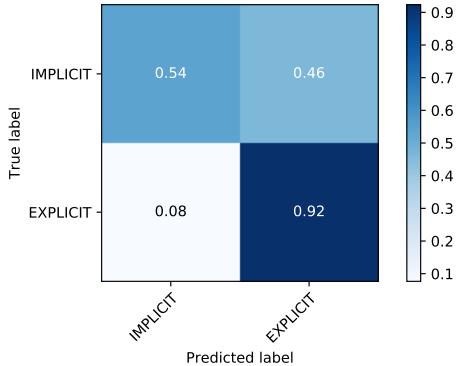


Figure 4: Normalized confusion matrix for the implicit/explicit classification subtask.

Our approach builds on a BERT model pre-trained on a large German-language corpus. To this end, we fine-tuned the model on the labeled task-specific training datasets and refrained from any feature engineering or sophisticated pre-processing. The evaluation results on the test data match the results on our validation data and the achieved macro-average F1 score beats the baseline by up to 26 percentage points. We showed that language models, such as BERT, can be successfully fine-tuned for offensive language detection for the German language.

One direction for future work on German BERT models is to find a better way for tokenization of words written in capitals. While capitalization certainly needs to be dealt with in German language models, our current model fails in recognizing words written in capitals. Another direction is the ensembling of multiple BERT models. We find that the predictions of models with differently initialized weights but trained on the same data varies. While ensembling these different models increases overall classification performance, it is unclear how this method can be leveraged best and where its limits are.

References

- Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 759–760. International World Wide Web Conferences Steering Committee.
- Kevin Coe, Kate Kenski, and Stephen A Rains. 2014. Online and uncivil? patterns and determinants of incivility in newspaper website comments. *Journal of Communication*, 64(4):658–679.
- Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the AAAI Conference on Web and Social Media*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, pages 1–16.
- Mark Hsueh, Kumar Yogeeswaran, and Sanna Malinen. 2015. leave your comment below: Can biased online comments influence our own prejudicial attitudes and behaviors? *Human Communication Research*, 41(4):557–576.
- Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting tweets against blacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 2227–2237. ACL.
- Julian Risch, Eva Krebs, Alexander Lser, Alexander Riese, and Ralf Krestel. 2018. Fine-grained classification of offensive language. In *Proceedings of GermEval (co-located with KONVENS)*, pages 38–44, September.
- Leonie Rösner, Stephan Winter, and Nicole C Krämer. 2016. Dangerous minds? effects of uncivil online comments on aggressive cognitions, emotions, and behavior. *Computers in Human Behavior*, 58:461–470.
- Nina Springer, Ines Engelmann, and Christian Pfaffinger. 2015. User comments: Motives and inhibitors to write and read. *Information, Communication & Society*, 18(7):798–815.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the International Workshop on Semantic Evaluation*, pages 75–86. ACL.

The HUIU Contribution for the GermEval Shared Task 2

**Melanie Andresen, Melitta Gillmann, Jowita Grala, Sarah Jablotschkin,
Lea Röseler, Eleonore Schmitt[†], Lena Schnee, Katharina Straka,
Michael Vauth, Sandra Kübler[‡], Heike Zinsmeister**

Universität Hamburg, [†]Universität Bamberg, [‡]Indiana University

{melanie.andresen, melitta.gillmann, sarah.jablotschkin}@uni-hamburg.de
 {Jowita.Grala, Lea.Roeseler}@studium.uni-hamburg.de
 {Lena.Schnee, Katharina.Straka}@studium.uni-hamburg.de
 {michael.vauth, heike.zinsmeister}@uni-hamburg.de
 eleonore.schmitt@uni-bamberg.de, skuebler@indiana.edu

Abstract

In this paper, we present the HUIU system (a collaboration of University of Hamburg and Indiana University) for the GermEval 2019 shared task 2, subtask 1 – the coarse-grained classification of tweets into the classes OFFENSE or OTHER. Our system uses linear SVMs with character n -grams ($5 \leq n \leq 10$), POS n -grams ($3 \leq n \leq 9$) and the tweet’s length in number of tokens as features. We obtain a macro-averaged F-score of 65.32 on the test data.

1 Introduction

In this paper, we report on the HUIU team’s submission to the *GermEval Task 2, 2019 - Shared Task on the Identification of Offensive Language*. Three subtasks were offered. Subtask I was a binary classification task and required discriminating offensive from non-offensive tweets. Subtask II consisted of a more fine-grained classification: Each of the offensive tweets had to be marked with one of the following labels: PROFANITY, INSULT, ABUSE. Subtask III required labeling the offensive tweets as explicitly or implicitly offensive. We participated in Subtask I, i.e., the detection of offensive language in binary classified twitter data.

Our contribution is the result of a class project conducted at the University of Hamburg. The authors participated in a 6-day compact class that provided an introduction to machine learning for linguistics and digital humanities, under the supervision of Kübler and Zinsmeister. Most of the participants had basic knowledge in programming, but no experience with machine learning. The class was structured to provide a practical introduction to machine learning. Therefore, the Shared Task offered a good opportunity to familiarize the participants with every step in the process of translating

a problem into a machine learning problem, deciding on a machine learning algorithm, specifying feature sets, extracting features, and training the machine learning algorithm. In addition to this task, the group also participated in GermEval 2019 Task 1 on hierarchical classification of blurbs (Andresen et al., 2019).

Using the python library scikit-learn (Pedregosa et al., 2011), we tested different models and features for the binary classification task of identifying offensive tweets. A bag-of-words approach which employs a linear SVM classifier using character n -grams combined with additional features yielded the best results.

The rest of the paper is structured as follows: We will briefly present the best systems of last year’s GermEval Shared Task as well as this year’s SemEval Shared Task in the section 2. In section 3, we will describe the experimental setup, i.e., the data of our Shared Task, how we preprocessed the tweets, which features we extracted, which classifier algorithm, implementation, and evaluation we used for our experiments. The best scores that we achieved during development are presented together with the final test scores in section 4. For our overall ranking in the Shared Task we have to refer to the summary published at the Workshop in Erlangen 2019. At the time of submitting this paper we did not have this information. In section 5, we will conclude our paper with a short summary and outlook on additional features and methods that we have not taken into account.

2 Related Work

The GermEval 2019 task on the Identification of Offensive Language is the second edition of the original task from 2018 (Wiegand et al., 2018). This year’s task is based on a different data set than

last year’s.¹ But parallel to 2018, Subtask I requires a binary classification system that discriminates between offensive and non-offensive tweets. In 2018 the team that reached the highest results (Montani and Schüller, 2018) extracted as features character n -grams, stemmed token n -grams, the TF/IDF scores of both feature classes, and word embedding vectors as features. The TF/IDF scores of the token n -grams proved to be their most important features, i.e., removing those from the model caused a large drop of the F1-score. Their classification system was an ensemble of supervised learning methods (Logistic Regression and Random Forests) implemented in scikit-learn (Pedregosa et al., 2011). Montani and Schüller (2018) reported worse results using deep learning models (LSTM, CNN, Convolution+GRU).

The GermEval shared tasks are the German equivalent of the SemEval-2019 Task 6 (OffensEval) Subtask A (Zampieri et al., 2019), which uses a data set consisting of English tweets. The data set is also remarkably larger than the one used in the GermEval 2018 task: It comprises more than 14,000 tweets (Zampieri et al., 2019), as compared to the approximately 5,000 tweets in the 2018 GermEval task. In contrast to the results reported by Montani and Schüller (2018), the best performing team at OffensEval Subtask A used a deep learning model – BERT (Devlin et al., 2018), based on bidirectional training of the attention model Transformer (Liu et al., 2019).

As discussed in the introduction, our goal of participating in the Shared Task was to acquire a basic understanding of machine learning in the setting of a compact introductory course. Therefore, we chose a common and easy to adapt SVM algorithm with a number of features, as described in the following section, and did not take into consideration prior more elaborated attempts at approaching this specific machine learning problem.

3 Experimental Setup

3.1 Data Set

We used the training and test data sets provided by the Shared Task, which consisted of 3,995 manually annotated tweets². Each tweet was labeled either as OFFENSE or as OTHER. On the second

¹The data set from 2018 was also available though.

²As mentioned above, the data set from 2018 was also available. We decided against using this additional set since we did not know if the data were different in distribution.

	OFFENSE	OTHER	sum
train	1,141	2,455	3,596
dev	146	253	399
sum	1,287	2,708	3,995

Table 1: Distribution of OFFENSE and OTHER in our training and development splits.

annotation level, each of the tweets of the category OFFENSE was marked with one of the following more fine-grained labels: ABUSE, PROFANITY, INSULT. Subtask I, in which we participated, did only take the binary coarse-grained labels into account. Examples (1) to (3) show example tweets from the annotated data set provided by the Shared Task. The annotation guidelines can be found online in the repository of the Shared Task 2018³.

- (1) @DrDavidBerger Die wirklichen Rassisten sitzen in der GroKo und bei den Grünen
(@DrDavidBerger The real racists are sitting in the GroKo and in the Green Party)
OFFENSE ABUSE
- (2) Sein Charakter war ihm wichtiger anstatt als billige Nute für Korrupte Regierungen zu arbeiten .Er hat das Leben begriffen
(‘His character was more important to him instead of working as cheap whore for corrupt governments .He understands life’)
OFFENSE PROFANITY
- (3) @de_sputnik Eine Weltherrschaft führt zum Krieg bis zum bitteren Ende
(@de_sputnik World domination leads to war to the bitter end)
OTHER OTHER

In order to optimize our system, we split the provided training set into 90% for our actual train(ing) set and 10% for our dev(elopment) set by taking every tenth instance for development. Table 1 shows the distribution of tweets and coarse-grained labels in the train and dev set respectively. For the final submission, we trained the system on the complete training set.

3.2 Extracted Features

For preprocessing, we tokenized and part-of-speech (POS) tagged the data. We used the python

³<https://github.com/uds-lsv/GermEval-2018-Data/blob/master/guidelines-iggsa-shared.pdf>

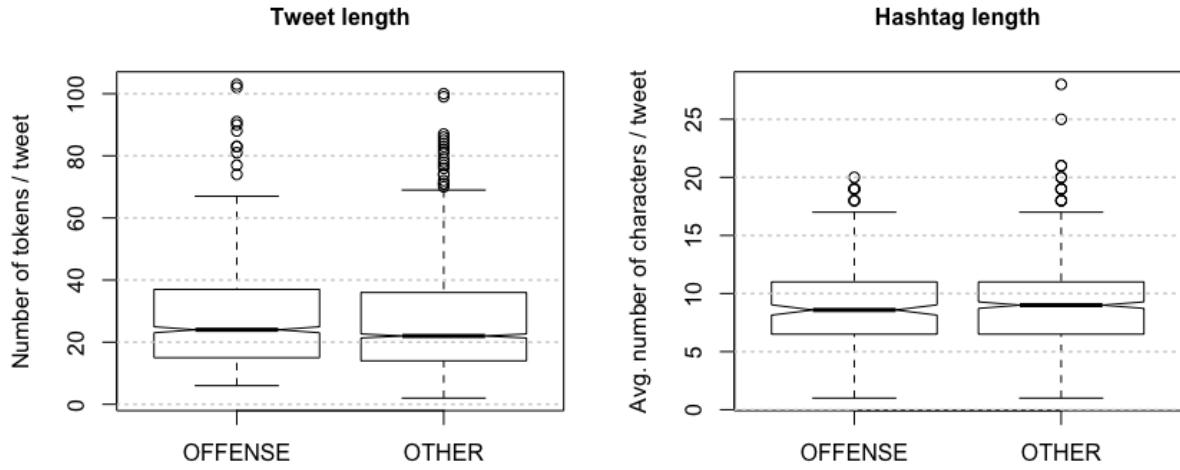


Figure 1: Boxplots of the distributions of tweet and (average) hashtag lengths per tweet in the 2019 training data (all tweets: n=3,995, tweets w/ hashtags: n=1,024; medians are marked by bold lines and notches).

implementation⁴ of twokenizer, a tokenizer especially designed for twitter data (Owoputi et al., 2013). For POS tagging, we employed *TnT* (Brants, 1998), trained on the Tübingen Treebank of Written Language (Tüba-D/Z) (Telljohann et al., 2006), version 10, assigning STTS labels (Schiller et al., 1999).

As basic features, we extracted token and POS n -grams for bag-of-words approaches of various classifiers, see Section 3.3 for the description of the classifiers.

We found that for our final system the bag-of-words approach was most effective when using character n -grams of the tokens (crossing word boundaries) combined with POS n -grams. For the tokens, the best results were achieved with a range of 5-10 characters; for the POS tags, a range of 3-9 words led to the best results.

In order to identify additional features that help to improve our model, we extracted further textual features per tweet. First, we counted the number of tokens per tweet as well as the number of @’s and #’s. Our hypothesis was that the emotional language of offensive tweets differed from other tweets in length and in the number of addressing terms and hashtags. In addition, we tested the length of hashtags because we assumed that in emotional tweets writers tend to use hashtags consisting of longer phrases or even full sentences.

⁴<https://github.com/myleott/ark-twokenize-py>

Therefore, we also determined the mean length of hashtags. Since we assumed that offensive tweets might be characterized by a specific use of punctuation marks and their combinations, we counted the occurrences of the following stand-alone punctuation marks . , ! ? as well as sequences of more than one of the same or different of these punctuation marks. We added an n -gram analysis of the (unicodes of) emojis extracted from the data, which we tested on character as well as on word basis.

Figures 1 and 2 illustrate the distribution of most of these features in the 2019 training data. In Figure 1, the boxplot on the left-hand side shows that offensive tweets are in fact significantly longer on average than other tweets.⁵ This does not hold true for the average lengths of the strings starting with #, see the boxplot on the right-hand side.⁶ Also the number of hashtags per tweet is on average slightly higher in other tweets than in offensive ones contrary to our expectations—if there are any hashtags at all, see the barplot for hashtag frequency in Figure 2.

With respect to addressing expressions, Figure 2 shows that offensive tweets tend to have in fact slightly more such elements than other tweets.⁷

⁵Tweet length: OFFENSIVE: median=24.00, mean=26.58; OTHER: median=22.00, mean=25.67; Wilcoxon rank sum test: W=1827400, p<0.05.

⁶Hashtag length: OFFENSIVE: median=8.58, mean=8.88; OTHER: median=9.00, mean=8.85; Wilcoxon rank sum test: W=80980, p=0.9021.

⁷Addressing expressions: OFFENSIVE: median=1,

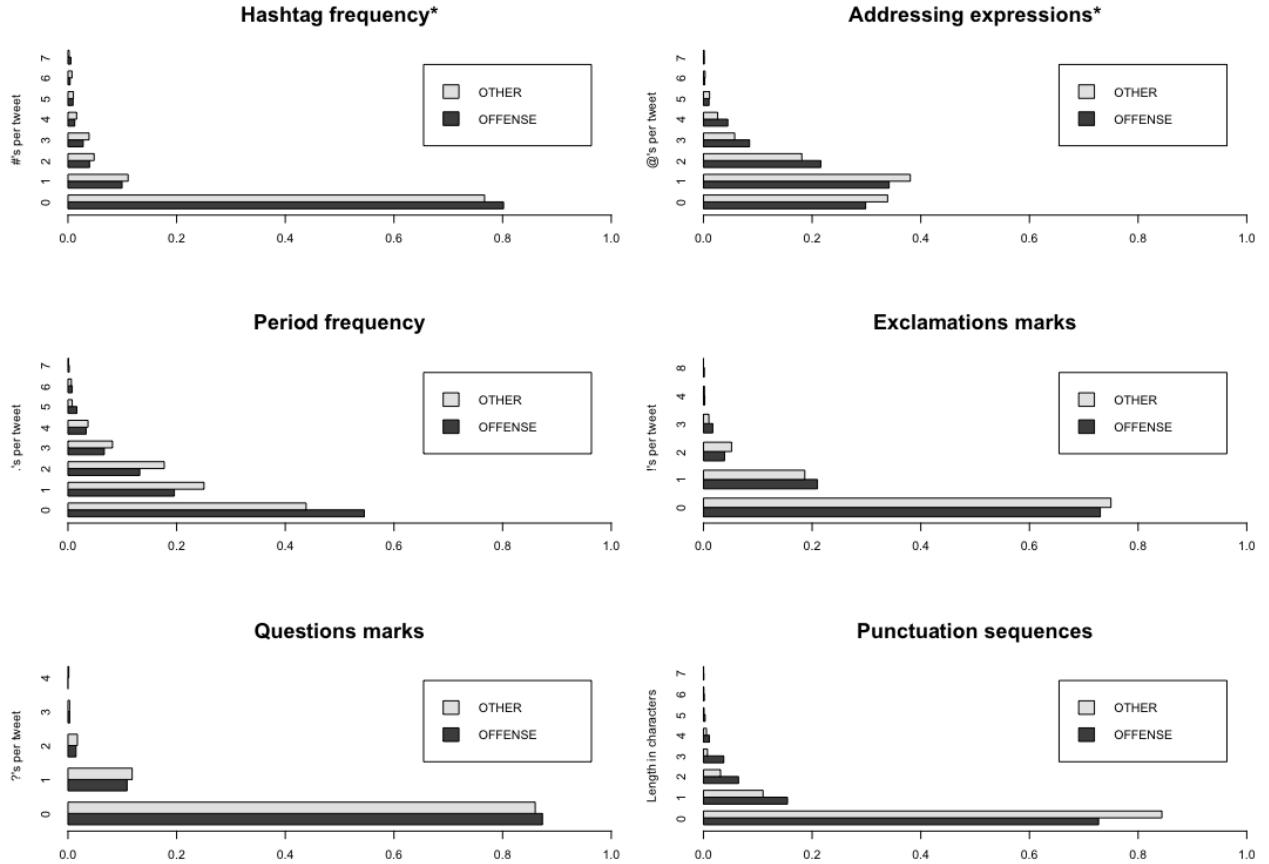


Figure 2: Additional feature distributions in the 2019 training data ($n=3,995$, cf. Table 1). X-axes: relative frequency of tweets. *) Numbers per tweet are cut off for better display: $y < 8$.

The same holds for sequences of punctuation marks (see bottom right of Figure 2). In contrast to addressing expressions, punctuation sequences are sparse in the data: About 73% of the offensive tweets and 84% of the other tweets do not contain any sequence of punctuation. The other barplots in Figure 2 show that data sparseness also holds for the distributions of exclamation marks and question marks, slightly less for periods which are on average more frequent in other tweets than in offensive ones.⁸

Among all these additional features, only the hashtag-related features did not improve our original results based on character and POS n -grams. We will present detailed results in Section 4.

mean=1.69; OTHER: median=1, mean=1.52; Wilcoxon rank sum test: $W=1889000$, $p<0.001$.

⁸Period frequency: OFFENSIVE: median=0, mean=0.94; OTHER: median=1, mean=1.08; Wilcoxon rank sum test: $W=1574100$, $p<0.001$.

3.3 Methodology

We used the machine learning library scikit-learn (v0.20.1) (Pedregosa et al., 2011) for Python (v3.7.1) and selected the Support Vector Classifier as our model. We achieved best results on our development data with a linear model default settings.⁹ Additional experiments with the Random Forest Classifier, including grid search for parameter tuning, did not yield better results.

3.4 Evaluation

For evaluation, we used the scorer provided by the shared task.¹⁰ It reports accuracy as percent correct, precision and recall for each subset (OFFENSE, OTHER), and the macro-averaged F1-score which is the harmonic mean of the results of the two subsets.

⁹Tests with the constructor option ‘probability =True’ yielded identical results with slower performance. For details see <https://scikit-learn.org/stable/modules/svm.html#svm-kernels>.

¹⁰<https://projects.fzai.h-da.de/iggsa/evaluation-tool/>

	Accuracy			OFFENSE			OTHER			Average		
	Perc.	corr.	total	P	R	F	P	R	F	P	R	F
dev	76.94	307	399	84.62	45.21	58.93	75.08	95.26	83.97	79.85	70.23	71.45
test	72.58	2,200	3,031	64.76	31.44	42.33	74.02	91.95	82.02	69.39	61.69	65.32

Table 2: Results on the development set and on the final test set (Average= macro-averaged score, Perc.= percent correct, corr.= number of correct tweets, total= number of all tweets, P=precision, R=recall, F=F1-score).

System	Accuracy		OFFENSE			OTHER			Average		
	Perc.	corr.	P	R	F	P	R	F	P	R	F
char (= baseline)	76.19	304	84.00	43.15	57.01	74.38	95.26	83.54	79.19	69.20	70.27
char #/tw	75.94	303	82.89	43.15	56.76	74.30	94.86	83.33	78.60	69.21	70.05
char #length	76.19	304	84.00	43.15	57.01	74.38	95.26	83.54	79.19	69.20	70.27
char POS	76.44	305	84.21	43.84	57.66	74.61	95.26	83.68	79.41	69.55	70.67
char ./tw	76.44	305	84.21	43.84	57.66	74.61	95.26	83.68	79.41	69.55	70.67
char !/tw	76.44	305	83.33	44.52	58.04	74.77	94.86	83.62	79.05	69.69	70.83
char pu-seq/tw	76.44	305	83.33	44.52	58.04	74.77	94.86	83.62	79.05	69.69	70.83
char @/tw	76.44	305	83.33	44.52	58.04	74.77	94.86	83.62	79.05	69.69	70.83
char ,/tw	76.69	306	84.42	44.52	58.30	74.84	95.26	83.83	79.63	69.89	71.06
char ?/tw	76.69	306	84.42	44.52	58.30	74.84	95.26	83.83	79.63	69.89	71.06
char w/tw	76.94	307	84.62	45.21	58.93	75.08	95.26	83.97	79.85	70.23	71.45
char w/tw ,/tw	76.94	307	84.62	45.21	58.93	75.08	95.26	83.97	79.85	70.23	71.45
char POS w/tw	76.94	307	84.62	45.21	58.93	75.08	95.26	83.97	79.85	70.23	71.45
char ,/tw ?/tw	76.69	306	84.42	44.52	58.30	74.84	95.26	83.83	79.63	69.89	71.06
char w/tw @/tw	76.69	306	83.54	45.21	58.67	75.00	94.86	83.77	79.27	70.03	71.22
char POS w/tw ./tw	75.94	303	82.05	43.84	57.14	74.45	94.47	83.28	78.25	69.15	70.21
char POS w/tw @/tw	76.44	305	83.33	44.52	58.04	74.77	94.86	83.62	79.05	69.69	70.83
all features	74.94	299	80.26	41.78	54.96	73.68	94.07	82.64	76.97	67.93	68.80

Table 3: Results of the ablation study and a model with all features (on the development set, n=399); best results in bold face (char=character n-grams, pu-seq=punctuation sequence, w= tokens, /tw= per tweet, Average= macro-averaged).

Averaging this way makes sure that in unbalanced settings, in which one subset is much larger than the other, the results on the larger subset do not obliterate the results on the smaller one. In case of the training and development data, the subset OTHER was much larger than the subset OFFENSE, cf. Table 1.

We optimized our system for the macro-averaged F1-score on our development set, since this score was the official ranking function in the shared task.

4 Results

We submitted one set of results (based on one run), obtained by one of the systems that had the best results on our development set: a linear SVM using character n -grams, POS n -grams, and tweet length as features (= system *char POS w/tw* in Table 3).

4.1 Official Shared Task Results

Our best system on the development set achieved a macro-averaged F1-score of 65.32 on the shared task’s test data, see Table 2.

Overall, the system did not generalize well to the final test data. We observe a loss of about 6 points in macro-averaged F1-score from 71.45 on the development data to 65.32 on the test data. The main decrease is due to a loss of about 20 points in precision on the OFFENSE class, followed by about 14 points in recall. The effect on the OTHER class is much smaller with only about 3 points loss in the F1-score from 83.97 to 82.02.

4.2 Ablation Study

We tested the best bag-of-words setting (character n -grams of size 5-10 and POS n -grams of size 3-9) with different additional feature combinations. Table 3 shows the results including some identical performances for the sake of completeness. The version with all features includes the following features in addition to the bag-of-words features (all measured per tweet): *number of tokens* (w/tw), *number of @’s* (@/tw), *number of #’s* (#/tw), *max_length of hashtag* (#-length), *mean_length of hashtags*, *number of commas* (,/tw),

number of periods (. /tw), number of exclamation marks (!/tw), number question marks (?/tw), punctuation sequences longer than one (pu-seq/tw).

Using only character n -grams provides a solid baseline (*char*). Adding the number of hashtags per tweet decreased the results slightly (*char #/tw*). Adding other individual features increased the overall result. POS n -grams improved precision in both classes marginally (*char POS*). The three best combinations used the number of words per tweet (*char w/tw*) as a feature. However, they outperformed models with other feature combinations only slightly. The difference was mostly due to improved recall of offensive tweets (of about 1.4 points in comparison to the model based on characters and POS n -grams only). We submitted only the best system combining character information with tweet length and POS information (*char POS w/tw*), hoping that POS n -grams can capture some generalizations. It differed in the annotation of the dev set in two tweets from the other two best performing systems (all three systems got these tweets wrong). The other two best performing variants (*char w/tw* and *char w/tw ,/tw*) yielded the exact same annotation results. The comma frequency (,/tw) does not seem to add additional information which is not also contained in tweet length.

The full version performed systematically worse than systems with fewer features. This could be an effect of overfitting.

We also tested adding unicode strings of emojis as n -gram features which did not effect the results in a positive way (not documented in Table 3).

In addition, we experimented by replacing all @-strings with a placeholder *NE* on the token level by keeping the original length information. The idea was to reduce overfitting on individual users. The results on our dev set were discouraging (macro-averaged F1-score using *char POS w/tw*: 67.46), so we did not further pursue this approach.

5 Conclusion and Future Work

By participating in the GermEval Shared Task in the setting of a compact introductory course we learned how to conduct the basic steps that are necessary when approaching a machine learning problem: From choosing a model to setting the parameters to extracting features from the data set and implementing them in the algorithm. For our final classifier, we used an SVM algorithm and optimized the system using several features of which

character n -grams and the length of the tweets with and without POS n -grams proved to be most effective.

We obtained an F-score of 65.32 on the test data. In future experiments, the score could possibly be improved by a larger training set and selecting more elaborate features: Montani and Schüller (2018), for example, obtained a high F-score in the GermEval 2018 Shared Task on the Identification of Offensive Language making use of the TF/IDF scores of token n -grams. They calculated the TF/IDF for each n -gram within each class (i.e. OFFENSE and OTHER) and created a feature that contained only those TF/IDF scores with a document frequency within a certain range (determined by a grid search). Thereby, they reduced the token n -gram counts to only those n -grams that are important for one of the classes.

Another promising source for potential feature extraction could be the emojis in the tweets with which we only did preliminary tests. They are an important characteristic of twitter data and reveal valuable information about the author’s intentions or emotional state. A semantic annotation (i.e. positive, negative) of each emoji type, perhaps with the help of the emoji descriptions in the unicode table, would precede the feature creation. This annotation would have to be done partly manually because emojis can be represented not only by simple but also by complex unicode containing variation selectors.

References

- Melanie Andresen, Melitta Gillmann, Jowita Grala, Sarah Jablotschkin, Lea Röseler, Eleonore Schmitt, Lena Schnee, Katarina Straka, Michael Vauth, Sandra Kübler, and Heike Zinsmeister. 2019. The HUIU Contribution to the GermEval 2019 Shared Task 1. In *Proceedings of the GermEval 2019 Workshop*, Erlangen, Germany.
- Thorsten Brants, 1998. *TnT-A Statistical Part-of-Speech Tagger*. Universität des Saarlandes, Computational Linguistics, Saarbrücken, Germany.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv e-prints*, page arXiv:1810.04805, Oct.
- Ping Liu, Wen Li, and Liang Zou. 2019. NULI at SemEval-2019 Task 6: Transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of the 13th International*

Workshop on Semantic Evaluation, pages 87–91, Minneapolis, MN.

Joaquín Padilla Montani and Peter Schüller. 2018. TUWienKBS at GermEval 2018: German abusive tweet detection. In *Proceedings of the GermEval 2018 Workshop*, pages 45–50, Vienna, Austria. Austrian Academy of Sciences.

Olutobi Owoputi, Brendan O’Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390, Atlanta, GA.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Anne Schiller, Simone Teufel, Christine Stöckert, and Christine Thielen. 1999. Guidelines für das Tagging deutscher Textcorpora mit STTS. Technical report, Institut für maschinelle Sprachverarbeitung & Seminar für Sprachwissenschaft, Stuttgart & Tübingen, Germany.

Heike Telljohann, Erhard W. Hinrichs, Sandra Kübler, and Heike Zinsmeister. 2006. *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*. Seminar für Sprachwissenschaft, Universität Tübingen, Germany.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of the GermEval 2018 Workshop*, pages 1–10, Vienna, Austria.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 Task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, MN.

TUWienKBS19 at GermEval Task 2, 2019: Ensemble Learning for German Offensive Language Detection

Joaquín Padilla Montani

TU Wien

Institut für Logic and Computation
Favoritenstraße 9-11, 1040 Austria
jpadillamontani@gmail.com

Peter Schüller

TU Wien

Institut für Logic and Computation
Favoritenstraße 9-11, 1040 Austria
ps@kr.tuwien.ac.at

Abstract

The TUWienKBS19 system for German offensive language detection in the GermEval 2019 shared task is a stacking ensemble system. Five disjoint sets of features are used: token and character n-grams, relatedness to the, according to tf-idf, most important tokens and character n-grams within each class, and the average of the embedding vectors of all tokens in a tweet. Several base classifiers are trained independently on each of these features, yielding meta-level features which one maximum entropy model uses to perform the final classification. Our system achieved a macro-averaged F_1 -score of 76,80% on subtask I, and 51,86% on subtask II.

1 Introduction

We describe the TUWienKBS19 system that participated in the GermEval Task 2, 2019, Shared Task on the Identification of Offensive Language.

This task is relevant for supporting humans when they moderate online content. In the pseudo-anonymous environment of microposts, abusive language is easily produced by users and it is an important objective to prevent that such content is broadcast to a large number of readers.

Our system is based on a stacked architecture where a set of base level classifiers is trained on a set of five feature groups, and the resulting trained models are forwarded to a meta-level classifier that determines the final outcome of the prediction. This architecture and its training method builds upon the system (Padilla Montani and Schüller, 2018) we submitted to GermEval 2018. This 2018 system was in turn inspired by the ELECTION system (Eger et al., 2017).

This year’s system compares to its predecessor (Padilla Montani and Schüller, 2018) as follows:

- Feature extraction was kept as before, although several hyperparameters were readjusted using this year’s training data.
- The set of base level classifiers used has been broadened, while keeping training times low.

This paper is organized as outlined below. In Section 2 we give details about the competition subtasks and evaluation metrics. In Section 3 we describe tweet preprocessing and the features we used. In Section 4 we describe the machine learning models we used and the stacked predictor model and we describe how we trained this architecture. Section 5 describes our submitted runs and provides the official competition scores for each run and subtask. We conclude the paper in Section 6.

The source code of our system, i.e., feature computation, training, and classification, is available online.¹

2 Data and Subtasks

The GermEval Task 2, 2019, Shared Task on the Identification of Offensive Language² solicited the submission of systems to automatically classify German language microposts (in a Twitter dataset) with respect to their offensiveness. Such predictions are a valuable tool for assisting human moderators with the job of reducing the amount of hurtful, derogatory or obscene online content.

This year’s edition of the Shared Task consisted of three subtasks:

- Subtask I: coarse-grained classification into the two classes “OFFENSE” and “OTHER” (where “OTHER” means non-offensive).
- Subtask II: fine-grained classification into the four classes “PROFANITY”, “INSULT”, “ABUSE”, and “OTHER”.

¹<https://github.com/jpadillamontani/germeval2019>

²<https://projects.fzai.h-da.de/iggsa/>

- Subtask III: offensive tweets are further classified into the two subcategories “EXPLICIT” and “IMPLICIT”.

In each of the subtasks, the classes are mutually exclusive. For example, in subtask II the “PROFANITY” class does not contain any insults, “ABUSE” does not insult a single concrete person but a whole group of people and is also abusive in a way that is not simply “PROFANITY”. For further details see also the annotation guidelines (Ruppenhofer et al., 2018).

The submitted systems are evaluated according to the macro-averaged F_1 -score of their predictions, i.e. each class contributes equally to the final score, independent from the number of samples in the class.

Our team participated in subtasks I and II. For these two subtasks, the training set contains 12536 tweets, where 8359 are marked as “OTHER” and the remaining ones as “OFFENSE”. Of the offensive tweets, 2305 are marked as “ABUSE”, 1601 as “INSULT”, and 271 as “PROFANITY”.

3 Features

We implemented feature extraction using the libraries *scikit-learn* (Pedregosa et al., 2011) for tf-idf computations, *NLTK* (Bird et al., 2009) for tokenization and stemming, and *gensim* (Řehůřek and Sojka, 2010) for managing precomputed word embeddings.

3.1 Preprocessing

Our preprocessing approach first removes all handles (@username) and replaces the special characters “#-_;:/><>&” and line break characters by spaces. The substring “s” (as in “geht’s”) is also replaced by a space.

For tokenization we used *NLTK*’s *TweetTokenizer* with `reduceLen=True`. This parameter means that repetitions of the same character are shortened to at most three letters (e.g., “coooooo” is normalized to “coool”).

For features for which we applied stemming, *NLTK*’s *GermanStemmer* was utilized.

Table 1 gives an overview of the groups of features we used and the type of preprocessing used in each case. We describe each feature group in the following.

Special Preprocessing indicates which additional preprocessing was done beyond handle removal, special character replacement and tokenization. For

creating character-level features we concatenated (Join in Table 1) the resulting tokens with spaces into one string for extracting character-level n-grams, i.e. we always used the tokenizer (even for character-level features) to make use of its `reduceLen` feature.

3.2 Character and Token N-Gram Features

The feature groups CNGR and TNGR are similar, so we describe them together. Both operate on a lowercased version of the input, and TNGR additionally performs stemming on each token.

CNGR extracts all character-level n-grams of length 3 to 7, while TNGR extracts all stemmed token-level n-grams of length 1 to 3.

In both cases, we performed tf-idf over all extracted n-grams. Only n-grams with a document frequency between 0.01 and 0.0002 at the token level, and only those with a document frequency between 0.02 and 0.0001 at the character level were kept (i.e., those that are rare enough to carry some signal, but frequent enough to have a potential to generalize over unseen data). The described document frequency thresholds were tuned by means of a grid search on a 90%/10% split of the training data, with the aim to maximize prediction scores of the base classifiers (see Section 4).

We used the tf-idf score of the relevant n-grams as input features (realized with *scikit-learn*’s *TfidfVectorizer*).

3.3 Word Embedding Features

We used a pretrained word2vec-style skip gram word embedding with 100 dimensions and window size 5, created from a large collection of German language tweets from the years 2013 to 2017 by Heidelberg University.³

For each tweet, we created 100 real-valued features by taking the average embedding of all tokens in the tweet, normalized to unit length with the ℓ^2 -norm.

Whenever a word embedding is required, i.e., for feature groups TIMP and EMB, and whenever the token is not in the vocabulary of the pretrained list of word embeddings, we performed a fallback operation. We searched for the largest prefix and the largest suffix of the token of length 3 or greater where we know a word embedding. If we find such

³http://www.cl.uni-heidelberg.de/english/research/downloads/resource_pages/GermanTwitterEmbeddings/GermanTwitterEmbeddings_data.shtml

Symbol	Name	Level	Special Preprocessing	Word Embeddings
CNGR	Character N-Grams	C	Lowercase + Join	-
CIMP	Important N-Grams	C	Join	-
TNGR	Token N-Grams	T	Lowercase + Stemming	-
TIMP	Important Tokens	T	-	min/max cos distance
EMB	Word Embeddings	T	-	average

Table 1: Groups of features used for classification. Handle removal, special character replacement and tokenization is used for all features. C and T stand for character and token level, respectively.

Subtask	m	Feature	k
I	2	CIMP	2000
	2	TIMP	2500
II	4	CIMP	500
	4	TIMP	1000

Table 2: Number of important types selected for each subtask and feature group.

affixes with embeddings, we use the embeddings of these affixes as if they were separate tokens in the tweet.

As an example, the word “Nichtdeutsche” (non-Germans) in the dataset does not exist in some pretrained word embedding models, so we encounter an OOV (out-of-vocabulary) exception. Our method would use as a fallback two word embeddings for affixes “Nicht” (not) and “deutsche” (German+Adj) because both affixes are present in the word embedding model. This fallback significantly reduced the number of OOV exceptions when extracting these features.

3.4 Important N-Gram and Token Features

These two groups of features are based on the same idea: to perform tf-idf over the whole dataset, select the k most important types relative to each of the m classes ($m = 2$ in subtask I, $m = 4$ in subtask II). We determine importance by ranking features according to their average tf-idf value in all documents in the respective class. Based on the resulting list of $k \cdot m$ most important type/class combinations we create a feature for each $k \cdot m$ combination. For CIMP each type is a character n-gram, while for TIMP each type is a token. Intuitively this selects the most distinguishing types per category.

Table 2 shows the number of important types selected for each subtask and each feature group. These values were adjusted with a grid search on

a 90%/10% split of the training data in order to maximize prediction scores of the base classifiers (see Section 4).

So far we have only discussed how important types are selected. We next describe which features are generated from these important types.

For TIMP, for each important type t in a tweet we obtain its word embedding \vec{t} and compute the maximum and the minimum cosine distance from \vec{t} to all other embeddings of other types in the same tweet. We use the same OOV-fallback described in Section 3.3. This yields a minimum and a maximum feature for each important type and each class: $2 \cdot k \cdot m$ real features for each tweet.

For CIMP we have no embedding information, therefore we create for each important type t a Boolean feature that indicates whether t is contained in the tweet or not. This yields a feature for each important type and each class: $k \cdot m$ Boolean features for each tweet.

By creating a set of features for each class, we increase the signal that can be learned for the “PRO-FANITY” class in subtask II, since this class contains a very small set of samples.

4 Classification

Our system is a stacking ensemble which builds upon the system (Padilla Montani and Schüller, 2018) we submitted to GermEval 2018. That system in turn was inspired by the ELECTION system of Eger et al. (2017).

We implemented most of the classification using the library *scikit-learn* (Pedregosa et al., 2011) and refer to class and function names of scikit-learn in the following (unless explicitly stated otherwise).

4.1 Base Classifiers

For each subtask and each of the 5 feature groups discussed in Section 3, we independently trained a varying number of base classifiers, selected out of:

Subtask	Feature	Base Classifiers
I	CNGR	ete, etg, lre, mnb, gbm
I	TNGR	ete, etg, lre, mnb
I	CIMP	lre, mnb, gbm
I	TIMP	ete, etg
I	EMB	ete, etg, lre, gbm
II	CNGR	ete, etg, lre, mnb
II	TNGR	ete, etg, lre, mnb
II	CIMP	lre, mnb, gbm
II	TIMP	ete, etg
II	EMB	ete, etg, lre, gbm

Table 3: Base classifiers used for each subtask and each feature group.

- (ete) an ensemble of random forests trained on samples of the training set (Geurts et al., 2006) using information gain as criterion for scoring the sample splits (class `ExtraTreesClassifier` with `criterion=entropy`),
- (etg) another ensemble of random forests trained using Gini impurity for scoring sample splits (class `ExtraTreesClassifier` with `criterion=gini`),
- (lre) a MaxEnt model with balanced class weights (class `LogisticRegression`),
- (mnb) a multinomial naive Bayes classifier (class `MultinomialNB`), and
- (gbm) a gradient boosting model with decision trees as base learners from the library *LightGBM* (Ke et al., 2017) (class `LGBMClassifier`).

We incorporated in the ensemble the base classifiers from the above list which were able to achieve good individual cross-validation performance after fine tuning and were also relatively fast to train. Table 3 details, for each subtask and feature group, which of the base classifiers were used. We trained a total of 18 base level classifiers in subtask I, and 17 in subtask II.

Each base classifier was trained on 90% of the training data, and used to predict class probabilities on the remaining 10%. We performed this process 10 times in a cross-validation manner to obtain

predictions for all tweets in the training data. Furthermore, we then also trained each base classifier on the whole training data, and used this models to predict class probabilities for the test data. This process generates the meta-level features that are used by the meta classifier, as described in the following section.

4.2 Meta Classifier

For subtask I we generated 36 meta-level features per tweet, using the probabilistic class predictions of 18 base classifiers (two classes). In subtask II we have 68 meta-level features per tweet, according to the probabilistic class predictions of 17 base classifiers (four classes).

On these features and the known true classes of the training tweets we trained a maximum entropy model (class `LogisticRegression`). We used balanced class weights and fine tuned the *C* parameter for each subtask using stratified cross validation, i.e. ensuring stable class ratios in each fold.

5 Submission

We submitted three runs for subtask I and three runs for subtask II, named `TUWienKBS19_coarse_#.txt` and `TUWienKBS19_fine_#.txt`, respectively, where # is the run number (1, 2 or 3).

Run 2 corresponds to the full ensemble system as described in the previous sections. Run 1 differs from run 2 in that the CIMP features (and the associated base classifiers) were disabled, since in our pre-competition testing of the ensemble using cross validation we obtained the best results when not using these features. Training these ensemble systems takes around 10 minutes for each subtask and for each run, on a regular desktop computer.

Run 3 is a lightweight single model, namely a MaxEnt model with balanced class weights (class `LogisticRegression`) trained on our best performing group of features: character level n-grams (CNGR). This run only takes a few seconds of training per subtask.

Table 4 shows the official macro-averaged F_1 -score on the testing data for each of our submitted runs.

6 Conclusion

We presented the TUWienKBS19 submission to the GermEval Task 2, 2019, Shared Task on the Identification of Offensive Language. We submitted runs

Subtask	Run	F_1 -score
I	1	76,80
I	2	76,75
I	3	71,42
II	1	51,23
II	2	51,86
II	3	46,94

Table 4: Official results of our system runs.

for subtask I (binary classification) and subtask II (fine-grained classification). Our approach used a stacking ensemble system which was based on our submission from last year’s shared task. We utilized five groups of features, some operating at the token level and some at the character level, and we also made extensive use of pretrained word embeddings.

Our system is built with a major challenge from this competition in mind: the evaluation mode in combination with the class imbalance in the training data. The competition evaluation uses macro-averaging, i.e., each class counts the same. At the same time, in subtask II, there is one class (“PROFANITY”) with only 271 tweets as samples within a training set which contains 12536 tweets. Our tuning efforts were focused on managing this class imbalance.

Acknowledgement

We thank the organizers of the competition. This work has received financial support from the European Union’s Horizon 2020 research and innovation programme under grant agreement 825619 (AI4EU).

References

- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O’Reilly Media Inc.
- Steffen Eger, Erik-Lân Do Dinh, Ilia Kutsnezov, Måsoud Kiaeha, and Iryna Gurevych. 2017. EELECTION at SemEval-2017 Task 10: Ensemble of nEurAL Learners for kEyphrase ClassificaTION. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval 2017)*, pages 942–946.
- Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine learning*, 63(1):3–42.
- Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems (NIPS 2017)*, 30:3146–3154.
- Joaquín Padilla Montani and Peter Schüller. 2018. TUWienKBS at GermEval 2018: German Abusive Tweet Detection. In *Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018)*, pages 45–50.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.
- Josef Ruppenhofer, Melanie Siegel, and Michael Wiegand. 2018. Guidelines for IGGSA Shared Task on the identification of offensive language. <http://www.coli.uni-saarland.de/~miwieg/Germeval/guidelines-iggsa-shared.pdf>.

RGCL at GermEval 2019: Offensive Language Detection with Deep Learning

Alistair Plum, Tharindu Ranasinghe, Constantin Orăsan, Ruslan Mitkov

Research Group in Computational Linguistics

University of Wolverhampton, UK

{a.j.plum, tharindu.ranasinghe, c.orasan, r.mitkov}@wlv.ac.uk

Abstract

This paper describes the system submitted by the RGCL team to *GermEval 2019 Shared Task 2: Identification of Offensive Language*. We experimented with five different neural network architectures in order to classify Tweets in terms of offensive language. By means of comparative evaluation, we select the best performing for each of the three subtasks. Overall, we demonstrate that using only minimal preprocessing we are able to obtain competitive results.

1 Introduction

The use of offensive language in open and public actions is a facet of the internet that calls for automatic detection for some kind of monitoring. The fact that people use language that can cause offence to other people is in no way a novel phenomenon. However, with the rise of online platforms such as Twitter, Facebook, Reddit and so on, along with the anonymity these platforms offer, offensive language can be viewed and read by millions of people in an instance. While the scale of the offence caused by such language can vary, it is clear that there is some language which causes offence to many people publicly. Therefore, it is desirable to be able to automatically detect the use of such language, in order to flag it and take further action.

Recently, efforts in the field of natural language processing (NLP) relating to the detection and classification of offensive language have been gaining attention. This is not only evidenced by an increase in offensive language datasets, but also a shift in approach from support vector machine classifiers to more modern neural networks (Schmidt and Wiegand, 2017). More evidence for the rising attention to offensive language detection lies in the fact that the topic has been featured at shared tasks.

The most prominent example is probably SemEval 2019 Task 6, which dealt with the identification and categorisation of offensive language in social media for English, and attracted around 800 teams with 115 submissions (Zampieri et al., 2019).

Another example of a shared task for offensive language is GermEval 2019 Task 2. It deals with the detection and classification of offensive language in German Twitter posts. The task itself is divided into three classification subtasks, with the first dealing with a binary classification, i.e. whether a tweet is offensive or not. Subtask II is a more fine-grained classification, including three levels of granularity, *profanity*, *insult* and *abuse*. These two subtasks were featured at GermEval 2018, meaning that data from two years was available (Wiegand et al., 2018). The final subtask, aimed at classifying implicit and explicit offensive language, was newly introduced this year.

This paper describes our submission to the GermEval shared task. We propose a simple, low-effort approach, with minimal data processing. We employ five different neural network architectures in order to perform the three classification subtasks, evaluate each network and select the three best performing architectures for our final submissions.

The paper is structured as follows. Section 2 describes the system that was submitted, split into a description of the dataset (Section 2.1), how the data was processed (Section 2.2) and the architecture of the classifier that was used (Section 2.3). Section 3 presents an analysis of the results of our evaluation of the five different architectures (Section 3.1), as well as of the final submission (Section 3.2). Finally, Section 4 offers some final remarks and a conclusion.

2 System Description

This section describes the shared task data, as well as the system that was used to classify the data. The dataset is grouped in two parts, and we use

minimal preprocessing in order to use the data. For classification, we used and compared five different neural network architectures suited to this task. Our implementation has been made available on Github.¹

2.1 Dataset

The data provided by the task organisers was split into subtasks I & II, and subtask III, which were in turn split into training and test sets. For subtasks I & II we concatenated the 2018 training set (Wiegand et al., 2018) with the 2019 training set, resulting in 9004 training instances for subtasks I & II. Subtask III was introduced for the first time this year, providing 1958 training instances.

Different tags are used for each subtask. The binary classification uses *OTHER* and *OFFENSIVE* to distinguish non-offensive and offensive text. Subtask II extends the last tag into *PROFANITY* indicating the use of offensive words without being aimed at anyone specific, *INSULT* which is like the previous but aimed at a specific person or entity, and *ABUSE* which combines the last two tags. The final subtask uses the *EXPLICIT* and *IMPLICIT* tags.

2.2 Text Preprocessing

As mentioned previously, the data preprocessing for this task was kept fairly minimal. More specifically, we perform only three specialised tasks for this data, followed by tokenisation. The tasks include removing usernames, converting to lower case and removing punctuation marks. The motivation for a minimal approach was mainly to demonstrate the effectiveness of the neural network architectures used. A secondary motivation is the portability to other languages, as the tasks carried out here should be relatively simple to perform in other languages. This does, however, highlight the importance of solid word embeddings, as the approach is completely reliant on them.

First, we completely remove all usernames from the texts, without inserting a placeholder. This is carried out quite simply by removing all strings beginning with the @ symbol, as this is how usernames are denoted on Twitter. The reasoning behind this step is mainly to remove noisy text, as it is highly unlikely that there would be any embeddings for the usernames. In addition, it stands to

reason that these usernames do not add any semantic meaning. Moreover, if, for instance, a majority of offensive tweets were written by one user, this could lead to bias in the system against one user. However, this task is targeted at offensive language, not offensive users.

Next, we convert the text to lower case letters. This step may seem counter intuitive for German, as capitalisation is used to differentiate nouns, which can cause a difference in meaning. For instance, *Rennen* can mean *the race* (noun), whereas *rennen* can mean *to run* (verb). Therefore, our first intuition was to keep capitalisation, however, after running with and without capitalisation our results indicated that all lower case text works better. We found that this leads to a smaller number of words for which no embedding is found, and higher precision and recall values. This finding is in line with previous findings using a similar approach based on neural networks (Stammbach et al., 2018).

Finally, we remove all kinds of punctuation marks and mathematical symbols. We insert a place-holder that indicates to the system that a punctuation mark would have been at this place. With this approach, we can handle each word in the same way, as leaving punctuation marks would lead them to be read together with a word, leading to no embedding being available.

2.3 System Architecture

After data processing each text is encoded using German fasttext (Mikolov et al., 2018) embeddings.² The encoded tweets are then classified by one of the neural network architectures. We evaluated five different neural network architectures for the classification tasks: pooled Gated Recurrent Unit (GRU) (Section 2.3.1), Long Short-Term Memory (LSTM) and GRU with Attention (Section 2.3.2), 2D Convolution with Pooling (Section 2.3.3), GRU with Capsule (Section 2.3.4) and LSTM with Capsule and Attention (Section 2.3.5).

The parameters of each architecture were optimised using 5-fold cross-validation considering binary cross entropy loss function and using adam optimiser (Kingma and Ba, 2015). The motivation for using 5-fold cross-validation was mainly the size of the data available for subtask III. Using a higher number of folds for cross-validation results in a low number of training and evaluation

¹<https://github.com/TharinduDR/GermEval-Task-2>

²<https://dl.fbaipublicfiles.com/fasttext/vectors-wiki/wiki.de.vec>

instances affects the performance of the architecture (Stone, 1974). We used the reducing learning rate on plateau technique when a deep learning architecture stopped improving. Deep learning architectures often benefit from reducing the learning rate by a factor once learning stagnates (Ravault and Gorti, 2018). We monitored validation loss and if no improvement was seen for 2 epochs, the learning rate was reduced by a factor of 0.6, since this value seemed to offer the best improvement.

These architectures were successfully applied to a number of classification tasks such as GRU for sequence labeling (Chung et al., 2014), GRU with capsule for toponym detection (Plum et al., 2019), and their success in these tasks inspired us to use them for the task at hand.

2.3.1 Pooled GRU

In this architecture, after the embedding layer, embedding vectors are fed to the bi-directional GRU (Chung et al., 2014) at their respective timestep. The bi-directional GRU-layer has 80 units. The final timestep output is fed into a max pooling layer and an average pooling layer in parallel (Scherer et al., 2010). After this, the outputs of the two pooling layers are concatenated and connected to a dense layer (Huang et al., 2017) activated with a sigmoid function. Additionally, there is a spatial dropout (Tompson et al., 2015) between the embedding layer and the bi-directional GRU layer to avoid over-fitting. This architecture has been discussed in (Kowsari et al., 2019) as a common architecture to perform text classification tasks.

2.3.2 LSTM and GRU with Attention

With this architecture, the output of the embedding layer goes through a spatial dropout (Tompson et al., 2015) and is then fed in parallel to a bi-directional LSTM-layer (Schuster and Paliwal, 1997) with self attention and a bi-directional GRU-layer (Chung et al., 2014) with self attention (Vaswani et al., 2017). Both the bi-directional LSTM-layer and the bi-directional GRU-layer have 40 units. The output from the bi-directional GRU-layer is fed into an average pooling layer and a max pooling layer. The output from these layers and the output of the bi-directional LSTM-layer are concatenated and connected to a dense layer with ReLU activation. After that, a dropout (Srivastava et al., 2014) is applied to the output and connected to a dense layer activated with a sigmoid function.

2.3.3 2D Convolution with Pooling

The fourth architecture takes a different approach than the previous architectures by using 2D convolution layers (Wu et al., 2018), rather than LSTM or GRU layers. The outputs of the embedding layers are connected to four 2D convolution layers (Wu et al., 2018), each with max pooling layers. All the 2D convolution layers were initialised with normal kernel initialiser. The outputs of these are concatenated and connected to a dense layer activated with a sigmoid function after applying a dropout (Srivastava et al., 2014). This architecture has been used in the Quora Insincere Questions Classification Kaggle competition³.

2.3.4 GRU with Capsule

Most of the previous architectures rely on a pooling layer. However, this architecture uses a capsule layer (Hinton et al., 2018) rather than pooling layers. After applying a spatial dropout (Tompson et al., 2015) the output of the embedding layer is fed into a bi-directional GRU-layer (Chung et al., 2014). The bi-directional GRU-layer has 100 units and was initialised with the Glorot normal kernel initialiser and orthogonal recurrent initialiser with 1.0 gain. The output is then connected to a capsule layer (Hinton et al., 2018). The output of the capsule layer is flattened and connected to a dense layer with ReLU activation, a dropout (Srivastava et al., 2014) and batch normalisation applied, and re-connected to a dense layer with sigmoid activation. This architecture has been used to detect locations within word windows (Plum et al., 2019).

2.3.5 LSTM with Capsule and Attention

The final architecture uses combination of a capsule layer (Hinton et al., 2018) and a self attention layer (Vaswani et al., 2017). After the embedding layer a spatial dropout (Tompson et al., 2015) is applied to the output, which is then fed into a bi-directional LSTM-layer (Schuster and Paliwal, 1997) with 80 units. The layer is initialised with the Glorot normal kernel initialiser and orthogonal recurrent initialiser with 1.0 gain. The output of the bi-directional LSTM-layer is fed into a capsule layer and to a self attention layer in parallel. Then each output of both capsule layers and the self attention layer goes through a DropConnect (Wan et al., 2013). They are concatenated before connecting

³<https://www.kaggle.com/c/quora-insincere-questions-classification>

to a dense layer with sigmoid activation. This architecture has been used in the *Jigsaw Unintended Bias in Toxicity Classification* competition.⁴

3 Results

This section presents the results of the evaluation of the five architectures, as well as the evaluation of the final submission. As outlined in the previous sections, we compare the performance of five different neural network architectures in order to select the best for each task. Therefore, an evaluation of each architecture was performed, the results of which are presented in Section 3.1. In Section 3.2 we present the results of the final submission as carried out by the organisers of the task. Although we submitted the runs of the three best performing systems, we only present the best performing here. The full results have been added to Appendix A.

3.1 Architecture Evaluation

This section describes how we selected the architectures for the final submissions in each subtask. For subtask I, all of the architectures were trained on the 2018 and 2019 training data. The architectures were evaluated on the 2018 test data. GRU with Capsule, 2D Convolution with Pooling and Pooled GRU had the best F1-scores with 0.743, 0.740 and 0.728, respectively.

Again, for subtask II all of the architectures were trained on the 2018 and 2019 training data, and evaluated on the 2018 test data. GRU with Capsule, 2D Convolution with Pooling and LSTM & GRU with Attention were selected for final submission, with F1-scores of 0.698, 0.695 and 0.684, respectively.

As subtask III was organised for the first time this year, we did not have 2018 training data for architecture training or 2018 testing data for evaluation. Nonetheless, for subtask III we used 20% of the available 2019 training data for the evaluation, and used the rest of the data for training. For subtask III, GRU with Capsule, Pooled GRU and 2D Convolution with Pooling were used for the final submission, as they had F1-scores of 0.887, 0.840 and 0.817, respectively.

It is interesting to note that the GRU with Capsule and 2D Convolution with Pooling architectures were always among the top three performing architectures.

⁴<https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification>

Subtask	P	R	F1	Acc.
I	79.49	67.94	73.26	77.96
II	58.64	36.53	45.02	72.35
III	65.55	72.55	68.87	80.11

Table 1: Results of the evaluation. All values are reported as percent.

3.2 Submission Results

This section presents the results of the evaluation of our submission. The evaluation was carried out by the task organisers, and at the time of writing the paper the results and rankings of other groups are not available. Therefore, we report only the evaluation provided to us by the task organisers. We report precision, recall and f-measure averaged overall for each classification subtask. Separate values for each group of each individual classification task are presented in the full results, as well as the results for the other two architectures. Table 1 shows the results of the evaluation of the best performing architecture, 2D Convolution with Pooling.

4 Conclusion

In this paper, we have presented our system for identifying offensive language in tweets. The system uses minimal preprocessing, and relies on word embeddings. We experimented with different neural network architectures in order to determine the most suitable for this task. Going by our evaluation, and the results provided by the task organisers, it is clear that 2D Convolution with Pooling scores highest overall.

While our system should be quite portable to other languages, due to non language-specific preprocessing, it is also clear that this aspect could potentially improve the performance of our system. Moreover, a detailed look into the results of the fine-grained classification of subtask II could yield good indications of how to improve the system for this kind of classification. Nonetheless, for future research we would like to see how well this system could perform in other languages on similar tasks.

References

- Junyoung Chung, Çaglar Gülcöhre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*.

- Geoffrey E. Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix capsules with EM routing. In *Proceedings of ICLR 2018*.
- Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. 2017. Densely Connected Convolutional Networks. In *Proceedings of IEEE CVPR 2017*.
- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of CoRR 2015*.
- Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura E. Barnes, and Donald E. Brown. 2019. Text Classification Algorithms: A Survey. *Information*, 10(4).
- Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of LREC 2018*.
- Alistair Plum, Tharindu Ranasinghe, Pablo Calleja, Constantin Orasan, and Ruslan Mitkov. 2019. RGCL-WLV at SemEval-2019 Task 12: Toponym Detection. In *Proceedings of SemEval-2019*.
- Mathieu Ravaut and Satya Gorti. 2018. Gradient descent revisited via an adaptive online learning rate. *arXiv preprint arXiv:1801.09136*.
- Dominik Scherer, Andreas C. Müller, and Sven Behnke. 2010. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *Proceedings of ICANN 2010*.
- Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*.
- Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681.
- Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.
- Dominik Stammbach, Azin Zahraei, Polina Stadnikova, and Dietrich Klakow. 2018. Offensive language detection with neural networks for GermEval task 2018. In *Proceedings of GermEval 2018*.
- Mervyn Stone. 1974. Cross-validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133.
- Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. 2015. Efficient object localization using Convolutional Networks. In *Proceedings of IEEE CVPR 2015*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of NIPS*.
- Li Wan, Matthew D. Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. 2013. Regularization of Neural Networks using DropConnect. In *Proceedings of ICML 2013*.
- Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval 2018*.
- Yunan Wu, Feng Yang, Ying Liu, Xuefan Zha, and Shaofeng Yuan. 2018. A Comparison of 1-D and 2-D Deep Convolutional Neural Networks in ECG Classification. In *Proceedings of IEEE Engineering in Medicine and Biology Society*.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). *CoRR*.

A Full Results

Subtask I		Accuracy		OFFENSE			OTHER			Average	
File	Percent	correct	total	P	R	F	P	R	F	P	F-Score
2	77.96	2363	3031	81.72	40.1	53.8	77.26	95.78	85.53	79.49	67.94
3	77.37	2345	3031	83.49	36.49	50.79	76.37	96.6	85.3	79.93	66.55
1	75.92	2301	3031	82.79	31.24	45.36	74.97	96.94	84.55	78.88	64.09
Subtask II		Accuracy		ABUSE			INSULT			Average	
File	Percent	correct	total	P	R	F	P	R	F	P	F-Score
2	72.35	2193	3031	49.04	25.5	33.55	52.84	20.26	29.29	36.53	45.02
1	71.56	2169	3031	45.66	25	32.31	54.3	17.86	26.89	36.74	44.66
3	72.45	2196	3031	48.53	24.75	32.78	50.28	19.39	27.99	35.39	39.09
Subtask III		Accuracy		PROFLANITY			OTHER			Average	
File	Percent	correct	total	P	R	F	P	R	F	P	F-Score
3	80.11	745	930	92.85	83.17	87.74	38.25	61.94	47.29	65.55	72.55
1	78.39	729	930	93.05	80.78	86.48	35.98	64.18	46.11	64.52	72.48
2	78.17	727	930	91.24	82.41	86.6	33.65	52.99	41.16	62.44	67.7

FKIE - Offensive Language Detection on Twitter at GermEval 2019

Theresa Krumbiegel

Fraunhofer FKIE

Fraunhoferstraße 20

53343 Wachtberg

theresa.krumbiegel@
fkie.fraunhofer.de

Abstract

We describe our submissions to the Shared Task on Identification of Offensive Language at GermEval 2019. We take part in all three subtasks, utilizing a Support Vector Machine (SVM) for subtasks 1 and 2, and a Long short-term memory (LSTM) neural net as well as a Convolutional neural net (CNN) for subtask 3. We obtained a macro-F1 score of 75.21 for subtask 1, 55.42 for subtask 2 and 64.20 for subtask 3 on a development set that was split from the overall training set provided by the organisers.

1 Introduction

The interest in systems that are able to classify offensive language on social media platforms such as Twitter has grown over the last years. Several scientific contributions deal with the development of such systems (cf.(Zampieri et al., 2019), (Hakimi Parizi et al., 2019)). As a consequence of this increased interest, critical voices can also be heard regarding the way in which offensive language is detected. Davidson et al. (2019), for example, report on finding racial bias in datasets that are used to train detection systems and Silva et al. (2016) state that designing an objective definition of hate speech is invariably difficult because of the complex context in which it needs to be integrated.

Within the frame of the GermEval shared task, offensive language is defined as "hurtful, derogatory or obscene comments made by one person to another" (Ruppenhofer et al., 2018). Three tasks are given regarding the detection of such language. The first of these is a coarse-grained binary classification task that aims at the general detection of offensive tweets. The categories OFFENSE and OTHER need to be assigned.

(1) @SusanBrenning In Sachen Verrat war die Kirche schon immer groß. OFFENSE

(2) @Doodoofist Das mach dir was zu essen Kamerad OTHER

In the second, fine-grained task, offensive tweets have to be further categorized into PROFANITY, INSULT and ABUSE where profanity depicts the least and abuse the most offensive class.

(3) Wie viel Oblaten muss ich denn jetzt essen bis ich ein Steak von Jesus zusammen hab?^ PROFANITY

(4) Sagt mal, kommt es nur mir so vor, oder ist das Staafunk Fernsehprogramm wirklich so scheiße? INSULT

(5) @YigidoYosi58 @Mesut_A @ntvde @ntv Bald seid ihr alle hier "Entsorgt"! ABUSE

The final task is binary and is directed at the distinction between EXPLICIT and IMPLICIT offense.

(6) @sozialromantik Eine Schande für Deutschland ist diese BOLSCHEWISMUS Regierung! EXPLICIT

(7) Was tut Ihr, wenn Ihr merkt, dass jemand grün wählt? IMPLICIT

2 Classification Approach

We used different system designs to solve the specified tasks. For subtask 1 and 2, we choose a SVM system that was created with the help of scikit-learn (Pedregosa et al., 2011). For subtask 3 we used a LSTM neural net as well as a CNN implemented with Keras (Chollet, 2015).

2.1 Data

To develop classification systems that can achieve satisfying results for the described tasks, a sufficient amount of training data is essential. The organisers provided two sets of training data, one for subtasks 1 and 2 and another one for subtask 3. The set for subtask 3 does not include additional tweets but categorizes OFFENSE examples taken from the first data set further into IMPLICIT and EXPLICIT. The data distribution of all training sets is presented in table 1.

Class	Tweets	%
<i>Subtask 1</i>		
Offense	4117	33.32
Other	8359	66.68
Total	12536	100
<i>Subtask 2</i>		
Profanity	271	2.16
Insult	1601	12.77
Abuse	2305	18.39
Other	8359	66.68
Total	12536	100
<i>Subtask 3</i>		
Implicit	259	13.23
Explicit	1699	86.77
Total	1958	100

Table 1: Training data distribution

To evaluate and optimize our systems during the training phase, we took a random sample of 20% of the provided data to form a development set for each task. The distribution of these samples can be found in table 2.

Class	Tweets	%
<i>Subtask 1</i>		
Offense	840	33.49
Other	1668	66.51
Total	2508	100
<i>Subtask 2</i>		
Profanity	41	1.63
Insult	321	12.80
Abuse	478	19.06
Other	1668	66.51
Total	2508	100
<i>Subtask 3</i>		
Implicit	47	15.61
Explicit	254	84.39
Total	301	100

Table 2: Development set distribution

2.2 Feature Description

For the classification with the SVM, a number of features were used during training. We combined these features into groups and assigned transformer weights to them.

Group	Feature
Sentiment	Sentiment Score
Character content	Character n-grams
Tweet content	Number of words Number of mentions Number of capital words Number of hashtags Number of emojis Number of exclamation and question marks Number of URLs
Pre-process	Removal of stop words Lemmatization

Table 3: Features used in subtask 1 and 2

The sentiment scores were extracted with the help of the Python module textblob-de¹. We used character n-grams that are weighted by their TF-IDF. Lemmatization was implemented with the Spacy lemmatizer². The described features were used for subtask 1 as well as subtask 2.

For the LSTM neural net and the CNN that were utilized in subtask 3, we merely pre-processed the data. No specific features were fed into the net. During pre-processing, we converted the text to lowercase and removed all punctuation and stop words. A German stop word list was acquired from the Python module stop-words³. Furthermore, we removed the line break token "lbr" and stemmed the text with the GermanStemmer by NLTK⁴. To be able to use the Tweets as input for the neural net, we created sequences out of the examples given, with a maximum length of 100. Shorter instances were padded.

3 Preliminary Results

We present our preliminary results. These results were obtained by testing our systems on self-compiled development sets that comprise 20% of the training data respectively. In addition, we report on 10-fold cross validation results.

¹<https://textblob.readthedocs.io/en/dev/>

²<https://spacy.io/api/lemmatizer>

³<https://pypi.org/project/stop-words/>

⁴https://www.nltk.org/_modules/nltk/stem/snowball.html

3.1 Subtask 1 and Subtask 2

We evaluated our systems with the described development sets. During the optimization phase we firstly experimented with different character n-gram ranges. The decision to start the evaluation with a search for the best performing character n-grams is based, among others, on findings of last year’s GermEval Shared Task on Identification of Offensive Language that show that character n-grams are rewarding features (Ruppenhofer et al., 2018). Detailed results for subtask 1 that were obtained during the development process can be found in tables 4 and 5.

Feature	Macro-F1 Development
Char 1-4 grams	67.62
Char 1-5 grams	69.58
Char 3-6 grams	69.65
Char 3-7 grams	69.38

Table 4: Character n-gram evaluation

Due to the minimal difference between the performance of character 1-5 grams and character 3-6 grams, we decided to continue the optimization of our system with both ranges. Character 1-5 grams outperformed character 3-6 grams regarding the classification of OFFENSE slightly (F1-score of 55.60 vs. 55.52) while character 3-6 grams exhibited better results for the OTHER class (F1-score of 83.79 vs. 83.55).

Feature Combination	Macro-F1 Development
1-5 grams + tweet content	68.85
1-5 grams + sentiment	69.27
1-5 grams + pre-process	74.40
1-5 grams + sentiment + pre-process	74.40
1-5 grams + sentiment + tweet content	68.94
1-5 grams + pre-process + tweet content	74.66
1-5 grams + sentiment + pre-process + tweet content	74.65
3-6 grams + tweet content	69.05
3-6 grams + sentiment	70.27
3-6 grams + pre-process	74.34
3-6 grams + sentiment + pre-process	74.44

3-6 grams + sentiment + tweet content	69.93
3-6 grams + pre-process + tweet content	73.91
3-6 grams + sentiment + pre-process + tweet content	74.31

Table 5: Feature evaluation

As can be seen in table 5, we achieved the best macro-F1 score, 74.66, when using character 1-5 grams in combination with pre-process and tweet content. It can be observed that even though the feature group tweet content does not improve the results in combination with character 1-5 grams alone, it *does* contribute to a higher macro-F1 score when used together with other features. We achieved a nearly equally high macro-F1 score of 74.65 with the combination of 1-5 grams and all other features. In the case of character 3-6 grams, a combination of all features except tweet content yields the best results, 74.44. We continue our training and evaluation process with the combination of character 1-5 grams, sentiment scores, tweet content and pre-process as well as the combination of character 1-5 grams, tweet content and pre-process.

As all feature groups were combined in a feature union, we were able to assign transformer weights to the different groups. The best performing combination was the following:

- Character content: 0.8
- Sentiment: 0.6
- Tweet content: 1.0
- Pre-process: 0.8

We obtained a final, highest macro-F1 score of 75.21. This score was obtained by using character 1-5 grams, pre-process and tweet content. The additional inclusion of sentiment scores yields a slightly lower macro-F1 score of 75.02.

	Precision	Recall	F1	Support
OTHER	80.68	90.89	85.48	1668
OFFENSE	75.83	56.79	64.94	840
macro avg	78.26	73.84	75.21	2508

Table 6: SVM results subtask 1

10-fold cross validation of the highest scoring system results in a mean macro-F1 score of 73.96.

For the second, fine grained subtask we implemented the same optimization process as for subtask 1. We found the best feature combination for the SVM to be character 3-6 grams and pre-process.

We achieved a macro-F1 score of 55.42 on our development set. Other combinations that yielded relatively high macro-F1 scores were character 3-6 grams, pre-process, tweet content and sentiment, (55.05), and character 3-6 grams, pre-process and sentiment (54.97).

	Precision	Recall	F1	Support
OTHER	78.15	92.63	84.77	1668
PROFANITY	60.71	41.46	49.28	41
INSULT	53.80	30.84	39.21	321
ABUSE	60.50	40.38	48.43	478
macro avg	63.29	51.33	55.42	2508

Table 7: SVM results subtask 2

We evaluated the best performing system with 10-fold cross validation and obtained a mean macro-F1 score of 46.18. The discrepancy to the results achieved when using a fixed development set can and should be attributed to variations in the data.

3.2 Subtask 3

For subtask 3, we trained and tested a LSTM neural net as well as a CNN. Even though we achieved the best results for subtask 1 and 2 with the SVM model, we obtained distinctly better results for subtask 3 when training and evaluating the corresponding data on a neural net (macro-F1 score of 0.46 with the SVM vs. 0.64 with a neural net). This was due to difficulties of predicting IMPLICIT tweets with the SVM. We obtained a very low F1 score of 02.82 for this category which impacted the final macro-F1 score negatively.

The input for the neural nets was pre-processed as described in subsection 2.2. With the LSTM neural net, we achieved a macro-F1 score of 64.20, the CNN produced a score of 64.06.

	Precision	Recall	F1	Support
EXPLICIT	89.63	85.04	87.27	254
IMPLICIT	36.67	46.81	41.12	47
macro avg	63.15	65.92	64.20	301

Table 8: LSTM neural net results

	Precision	Recall	F1	Support
EXPLICIT	88.89	88.19	88.54	254
IMPLICIT	38.78	40.43	39.58	47
macro avg	63.83	64.31	64.06	301

Table 9: CNN results

3.3 Submitted Results

The following files were submitted:

1. fkie_coarse_1.txt — SVM, character 1-5 grams, pre-process, tweet content
2. fkie_coarse_2.txt — SVM, character 1-5 grams, pre-process, tweet content, sentiment

3. fkie_fine_1.txt — SVM, character 3-6 grams, pre-process
4. fkie_fine_2.txt — SVM, character 3-6 grams, pre-process, sentiment
5. fkie_fine_3.txt — SVM, character 3-6 grams, pre-process, tweet content, sentiment
6. fkie_implicit_1.txt — LSTM
7. fkie_implicit_2.txt — LSTM
8. fkie_implicit_3.txt — CNN

For subtask 1 (coarse) we submitted one run (fkie_coarse_1.txt) which uses character 1-5 grams, pre-process and tweet content as features for the SVM and another run (fkie_coarse_2.txt) which in addition uses sentiment scores.

The first submission for subtask 2 (fkie_fine_1.txt) was obtained by using a SVM with character 3-6 and pre-process. For the second submission (fkie_fine_2.txt), we again used character 3-6 grams, pre-process and added sentiment scores. The third submitted run (fkie_fine_3.txt) uses all available features.

For subtask 3 (implicit), three runs were submitted. Two of these (fkie_implicit_1.txt, fkie_implicit_2.txt) include results obtained with the LSTM neural net. The other one (fkie_implicit_3.txt) presents the CNN results.

4 Discussion

In general, the binary classification systems yield better macro-F1 scores than the multi-class system. This was to be expected. The best performing system is the one that focuses on the simple distinction between offensive and not offensive tweets. This is also intuitive: categorizing offensive tweets into profanity, insult, abuse or explicit, implicit, requires more precise feature engineering.

A fine-grained classification is, in this case, additionally difficult as the annotation of some sub-categories is at times not coherent, e.g.:

(8) @KingGeorgVI @EngelGert Ich kann es nicht mehr sagen. Bild und Artikel sind verschwunden und ich habe es nicht gespeichert. Das Bild zum Tweet ist ebenfalls weg. Sorry. OFFENSE ABUSE

(9) @JuttaMBrandt @jouwatch Ich muss da nicht überlegen. OFFENSE INSULT

The examples above are annotated as OFFENSE even though they do not appear to be insulting or

abusive. Especially example (8), which is categorized as abuse, the *most* offensive class, does not include offensive content but rather conveys a factual and even apologetic tone.

We assume that the categorization of such instances is based on background knowledge that is not accessible to us and subsequently not accessible to the classification systems. Still, it is advisable to train the systems on the provided data set and therefore inevitably on examples that break ranks, to make them applicable to the test set.

Regarding subtask 3, it is instinctive that the detection of EXPLICIT instances can be achieved more easily than that of IMPLICIT ones. The term implicit as such can be defined as “capable of being understood from something else though unexpressed” (Merriam-Webster, 2011) which already hints at the problem that something that is not overtly expressed might be difficult to identify. The impact of this can be observed clearly in the results depicted in tables 8 and 9. In addition, the distribution of EXPLICIT and IMPLICIT tweets in the training data is skewed (86.77% EXPLICIT, 13.23% IMPLICIT). This complicates the eventual detection of IMPLICIT tweets in the test data.

5 Conclusion

We presented our submission to GermEval Task 2, 2019 - Shared Task on the Identification of Offensive Language. We described the generation and implementation of a SVM, a CNN and a LSTM neural net as well as feature engineering and pre-processing strategies that were used.

For future work in this area, some issues should be considered and, if possible, improved. The data set that was provided for the training of the systems should be more balanced with regard to the individual categories. Especially for subtask 3, the small number of IMPLICIT examples was problematic. In addition, it would be helpful if the data was annotated in a more consistent manner. A data set that is fully coherent will quite likely improve the performance of the classification systems in the end.

Acknowledgements

We would like to thank our colleague Albert Pritzkau for his invaluable help and insights during the process of solving the task of offensive language detection.

References

- Francois Chollet. 2015. Keras. <https://keras.io>. [Online; accessed 24-July-2019].
- Thomas Davidson, Debasmita Bhattacharya, and Ingmar Weber. 2019. Racial bias in hate speech and abusive language detection datasets. *CoRR*, abs/1905.12516.
- Ali Hakimi Parizi, Milton King, and Paul Cook. 2019. UNBNLP at SemEval-2019 task 5 and 6: Using language models to detect hate speech and offensive language. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 514–518, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Merriam-Webster. 2011. implicit. <https://www.merriam-webster.com/dictionary/implicit>. [Online; accessed 14-August-2019].
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Josef Ruppenhofer, Melanie Siegel, and Michael Wiegand. 2018. Overview. In *Proceedings of the GermEval 2018 Workshop, 14th Conference on Natural Language Processing (KONVENS 2018)*, pages 1–10.
- Leandro Silva, Mainack Mondal, Denzil Correa, Fabricio Benevenuto, and Ingmar Weber. 2016. Analyzing the targets of hate in online social media. In *Proceedings of the Tenth International AAAI Conference on Web and Social Media (ICWSM 2016)*, pages 687–690, Palo Alto, California, USA. AAAI Press.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

bertZH at GermEval 2019: Fine-Grained Classification of German Offensive Language using Fine-Tuned BERT

Tim Graf

University of Zurich

Institute for Computational Linguistics

Andreasstrasse 15, 8050 Zrich

tim.graf@uzh.ch

Luca Salini

University of Zurich

Institute for Computational Linguistics

Andreasstrasse 15, 8050 Zrich

lucalelio.salini@uzh.ch

Abstract

The bertZH system for abusive tweet detection in the GermEval 2019 competition is a neural classifier based on BERT (Devlin et al., 2018). We describe our submission runs for subtask 2 on fine-grained classification of tweets. We used the pretrained German language model from deepset.ai¹ implemented in pytorch and fine-tuned it to the data of the task, before then using it to train on the classification task. We also experimented with the pretrained multilingual BERT model from Google Research implemented in keras, but it resulted in a worse score than with the German model. We have found that a language-specific BERT model outperforms a multilingual model and that fine-tuning a BERT model to the tasks domain achieves a small gain in performance.

1 Introduction

It can be very useful for the user experience of a social media platform to sort out abusive content. But first one has to know what content can be declared as abusive in order to avoid false-positives. The goal of our deep neural network is to find this abusive content.

We developed our models as a part of a Text Mining course at the University of Zurich as a final work. We are two Bachelor students in Computational Linguistics.

This paper is organized as follows: in section 2 we will explain the details of the competition, especially task 2. Then, in section 3 we provide some details about the preprocessing of our pipeline. In the 4th section we present the architecture of our deep neural network and the background of BERT. In section 5 we describe the configuration of each

submitted run in detail and we finally present our results in section 6.

2 Competition Tasks

The GermEval 2019 Shared Task on the Identification of Offensive Language² focused on classification of German tweets with respect to their offensiveness. With the overwhelming amount of social media posts everyday, systems that can reliably detect profane language or harassment grow more important in assisting human moderators.

- Subtask 1: Coarse grained classification. This dataset was labelled with only two labels, namely OFFENSIVE and OTHER, where OTHER represents non-offensive tweets.
- Subtask 2: Fine grained classification. For this task, each sample of the dataset (which is the same as in subtask 1) is labelled with four labels: INSULT, PROFANITY, ABUSE and OTHER.

We only participate in subtask 2. The task is a multi-class classification problem, which means that each tweet is only labelled with a single label (e.g. an abusive tweet that uses profane language is only labelled ABUSE). The data is not uniformly distributed as the class OTHER has a frequency of 67.8%, while the others are quite under-represented (INSULT: 15.6%, ABUSE: 12.7%, PROFANITY: 3.8%). This usually makes it very difficult to learn automatically how to predict the under-represented classes - especially the class PROFANITY.

The evaluation metric is F1-score, hence it is important to have a good classification rate for every single class.

3 Preprocessing

Since tweets contain a lot of colloquial language and also hashtags or usernames or similar, we

¹<https://deepset.ai/german-bert>

²<https://projects.fzai.h-da.de/iggsa>

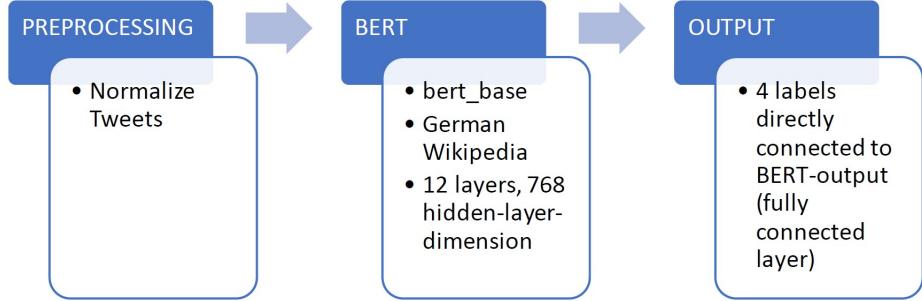


Figure 1: Top-level overview of our pytorch architecture.

needed to filter or normalize such occurrences. For that reason we used the German tokenizer SoMaJo³ and added some extra cleaning steps:

- Normalizing character repetitions: We replace characters which occur more than twice in a row with two of them ("coooooool" → "cool").
- Substituting usernames: Every username gets replaced with "@USER". We didn't cut out the whole username because it could be important for the classification if someone is mentioned.
- Removing special characters: We remove characters such as hashtags, newlines, line-breaks or underscores.

We also used scikit-learn (Pedregosa and others, 2011) for the train-test split during development.

4 Architecture

BERT (Devlin et al., 2018) has proven to be exceptionally effective in many downstream NLP-tasks including sentence classification. It has improved the state-of-the-art in several applications, and hence our goal was to implement BERT for the fine-grained classification task. However, training a BERT model from scratch is computationally very expensive and impossible to train on a single consumer-grade GPU, so we had to rely on the publicly available models. When we first started the project, the only available BERT model that was trained on German data was bert_multi_cased_L-12_H-768_A-12, a BERT model released by Google Research on GitHub⁴ that was trained on Wikipedia dumps in

104 different languages, of which one was German. In order to use the model in keras (Chollet and others, 2015), we followed Jacob Zweig's blogpost *BERT in Keras with Tensorflow hub*⁵. With this implementation, we could fine-tune the last n layers of the BERT transformer while connecting a 256-units Feed-Forward layer with dropout to the first generated token by BERT. This [CLS] token is a representation of the whole sequence and is the only component of BERT's output we use to perform the classification task (Devlin et al., 2018).

Later on in the project, we found that deepset released a BERT model to the public that was trained on German data exclusively (bert-base-german-cased)⁶, which was promising better results on several German tasks than the multilingual model by Google Research, including the GermEval 2018 Shared Task on the Identification of Offensive Language (Wiegand et al., 2018). The implementation in keras we described in the preceding paragraph relied on the model being available as a module on *TensorFlow Hub*, which was not the case for this model. Hence, we used the well-known pytorch (Paszke et al., 2017) implementation of BERT by the HuggingFace team⁷ and followed the blogpost *A Simple Guide On Using BERT for Binary Text Classification*⁸ by Thilina Rajapakse to be able to use the German model, and modified the code to suit the multiclass classification task. With the implementation by HuggingFace we were able to fine-tune the German model on to the tasks dataset using BERT's original language modeling tasks MLM

⁵<https://towardsdatascience.com/bert-in-keras-with-tensorflow-hub-76bc9417b>

⁶<https://deepset.ai/german-bert>

⁷<https://github.com/huggingface/pytorch-transformers>

⁸<https://medium.com/swlh/a-simple-guide-on-using-bert-for-text-classification-bbf041ac8d04>

³<https://github.com/tsproisl/SoMaJo>

⁴<https://github.com/google-research/bert>

(masked language modeling) and next sentence prediction before we trained the model to actually perform classification (Devlin et al., 2018). For classification, we then used the already provided `BertForSequenceClassification` model architecture without modifying it at all. We conducted some informal experiments on the following hyperparameters:

- Number of layers to fine-tune BERT (`keras`-implementation)
- rate of dropout (`keras`-implementation)
- Learning rate (both implementations)
- Number of epochs

5 Submitted Runs

5.1 Run 1

Run 1 was a submission that was trained on 11536 samples and evaluated on 1000 samples. It used the `pytorch`-implementation with the German BERT model and the following Hyperparameters:

Hyperparameter	Size
Learning rate	0.00002
# of epochs	5

Table 1: Set Hyperparameters of run 1.

5.2 Run 2

Run 2 was a blind submission, which we trained on all of the available 12536 samples, which means we did not know how well the system would actually perform. It was a `pytorch`-implementation using the German BERT model as well and used the following Hyperparameters:

Hyperparameter	Size
Learning rate	0.00002
# of epochs	5

Table 2: Set Hyperparameters of run 2.

5.3 Run 3

Our third submission was made with the `keras`-implementation and Google-Research’s multilingual model. Even though we were observing significantly worse performance using this model, we

were interested in how well this model would perform. This submission was trained with the following Hyperparameters:

Hyperparameter	Size
Learning rate	0.00002
# of epochs	3
# of fine-tuned layers	3
Dropout	0.5

Table 3: Set Hyperparameters of run 3.

5.4 Training Times

All of our experiments were conducted on a single RTX 2080ti GPU. The fine-tuning of the German BERT model took around 60 minutes for 3 epochs, and the training of the classification tasks for runs 1 and 2 took around 15 minutes. The `keras`-implementation of run 3 finished in 6 minutes. It is very impressive that using such powerful and large models is possible within very reasonable time-frames on consumer grade GPUs, and the practice of open-sourcing these large pretrained models should be applauded.

6 Results

We pre-calculated the F1-score for our different systems:

Features	F1	Diff
<i>run 2: with all data</i>	-	-
<i>run 1: German BERT + fine-tuning</i>	0.65	-
<i>(no submission): German BERT</i>	0.63	-0.02
<i>run 3: multilingual BERT</i>	0.53	-0.12

Table 4: Pre-calculated F1-scores of the models.

Features	F1	Diff
<i>run 2: with all data</i>	0.53	-
<i>run 1: German BERT + fine-tuning</i>	0.52	-0.01
<i>(no submission): German BERT</i>	0.50	-0.03
<i>run 3: multilingual BERT</i>	0.43	-0.1

Table 5: Final F1-scores of the models.

From these results it is obvious that a language specific BERT model improves the performance of a system. This should hold for any language. We also assume that our models are not yet saturated and that more training data would help achieve

Class	F1
<i>OTHER</i>	0.87
<i>ABUSE</i>	0.59
<i>INSULT</i>	0.54
<i>PROFANITY</i>	0.56

Table 6: F1-score distribution of the different classes for run 1: *German BERT with fine-tuning*.

an even higher score without any modification to the model, especially because more samples from the underrepresented classes should help the BERT model to get a better grasp of what for example makes a tweet an INSULT and not an abusive tweet. Another observation to point out is that fine-tuning a BERT model to task-specific data seems to improve the score even further. Hence, given enough training examples, BERT might be all you need.

7 Conclusion

After BERT has revolutionized the NLP-Community, we have applied it to the task of German offensive language detection. A common problem with neural approaches is that they usually require a larger amount of training data than more traditional machine learning approaches. However, large, pre-trained language models seem to model a language well enough so that even with a rather small dataset of 12536 they can be used to achieve impressive results. It was also very impressive to see that, even though PROFANITY made up only 3.8% of the training data, without any further data augmentation or oversampling, the BERT model did not face the problem of not predicting PROFANITY at all. Hence, our submission shows that relatively good results can be achieved without spending many resources on feature engineering or training large models, as fine-tuning existing released models does not take a lot of time.

Acknowledgments

We would like to thank deepset.ai and Google for making their BERT models available to the public. Furthermore, we thank our instructor Simon Clematide for his support and inputs.

References

François Chollet et al. 2015. Keras. <https://keras.io>.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Fabian Pedregosa et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language.