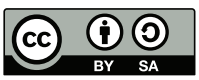
Text Analysis and Retrieval 2021

Course Project Reports

University of Zagreb

Faculty of Electrical Engineering and Computing



Publisher:

University of Zagreb, Faculty of Electrical Engineering and Computing

Organizers & editors:

Josip Jukić
Jan Šnajder

Reviewers:

Matija Bertović
Ivan Crnomarković
David Dukić
Sven Goluža
Josip Jukić
Mladen Karan
Antun Magdić
Patrik Marić
Zoran Medić
Dora Pušelj
Jan Šnajder
Martin Tutek

ISBN 978-953-184-274-7

Course website:

http://www.fer.unizg.hr/predmet/apt

Powered by:

Text Analysis and Knowledge Engineering Lab
University of Zagreb
Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
http://takelab.fer.hr

Preface

This is the eight booklet in a series of students' project reports from the Text Analysis and Retrieval (TAR) course, taught at the Faculty of Electrical Engineering and Computing (FER), University of Zagreb. TAR teaches the foundations of natural language processing and information retrieval, with a focus on practical applications, all while being in line with best practices in the area.

These outcomes are achieved through hands-on student projects, which are the central part of the course. Students that complete this course gain both practical and theoretical skills in data science and machine learning, with a strong focus on text data. Given the bewildering and ever-growing amount of information available today (in text form especially), such skills are invaluable to employers. We are happy and proud to supply our students with such an extra edge on the increasingly competitive job market.

This booklet represents the results of 18 projects, which are the work of 52 students. Most of the topics were adopted from recent workshops and shared tasks like SemEval and CLEF. This year we have had a very diverse set of topics, including irony detection, text-based personality assessment and author profiling, stress analysis in social media, detection of bullying traces, and offensive language detection, to name a few. With respect to methods, the trend of increasing popularity of deep learning is present for several years and continues this year, with most teams relying on some sort of deep learning to solve their task. We believe part of the reason for this is the increasing availability of deep learning libraries that are simple to use and the recent development of contextualized word embedding models that provide a high quality pre-trained neural word representations useful for almost any task. Some of the teams went well above and beyond by implementing, for example, a custom neural architecture coupled with a custom data augmentation technique.

As in the previous years, the project reports were written in the form of a research paper. The aim of this is to both teach the students to effectively present their results, as well as to give them a feeling of how actual scientific research works. To this end, in addition to writing a project report, the students also actively participated in several paper reading sessions, where scientific papers were discussed in groups. As students seldom get an opportunity to get involved in hands-on scientific research during their Master's programme, we felt this would be a valuable new experience for them. Similarly to previous editions of the course, our intuitions about this approach were confirmed by resoundingly positive student feedback.

Same as last year, this year's edition of TAR was held completely on-line due to the COVID-19 pandemic. While this required some organizational changes yet again, we believe we have managed to preserve the intended experience as much as possible under the circumstances. This was facilitated by considerable patience and understanding on part of the students. As the course organizers, we thank the students for demonstrating remarkable motivation and enthusiasm throughout the course. We are honored to have had the opportunity to work with them.

Josip Jukić and Jan Šnajder

Contents

Essays are a Fickle Thing

Lucija Arambašić, Miroslav Bićanić, Frano Rajič

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{Lucija.Arambasic,Miroslav.Bicanic,Frano.Rajic}@fer.hr

Abstract

Automatic classification of a person's personality based on a piece of text written by that person is an inherently difficult task, but its difficulty could increase depending on the dataset used. In this work, we explore the classification performance of many different machine learning models with various feature combinations when the dataset consists of stream-of-consciousness essays written by students. Despite achieving very good performance, we argue that such a dataset may not be ideal for personality trait classification.

1. Introduction

Personality is defined as the pattern of thoughts, feelings and behaviour specific to each individual. Personality is often characterized by using different personality traits. One of the most famous ways of describing personality is the Big Five model, which defines five fundamental personality traits: extroversion (EXT), neuroticism (NEU), agreeableness (AGR), conscientiousness (CON) and openness to experience (OPN).

Automatic personality assessment from text has numerous applications, ranging from job interviews to author profiling. Because of this, there was a need for an appropriate dataset. Social network platforms are a popular source for such data since posts on them often contain people's opinions and feelings. Park et al. (2014) built such a dataset using Facebook posts, while Gjurković et al. (2020) built their dataset using posts from Reddit. One popular dataset not obtained from social media is the *essays* dataset, in which stream-of-consciousness essays are matched with Big Five gold labeling of their authors.

In this paper, we analyze our attempt to perform personality trait classification of essay authors on the *essays* dataset. We employ several machine learning techniques, including both static and sequence-based models, utilizing various standard and hand-crafted features. Additionally, we experiment with data augmentation. Finally, we bring into question the quality and applicability of the *essays* dataset for personality trait classification.

2. Related Work

Some of the previous work regarding personality trait classification from text was done by the authors of the *essays* dataset. They extracted Linguistic Inquiry and Word Count (LIWC) features and used them to determine the link between the written essay and the authors Big Five personality traits (Pennebaker and King, 1999).

Similarly to us, Pizzolli and Strapparava (2019) trained their models on the *essays* dataset, but then used the model to classify the personality traits of characters in Shakespearean plays. It is important to note that this task does not have gold labels, so the performance was evaluated manually and subjectively.

More recent work on personality trait classification using the *essays* dataset is done in (Majumder et al., 2017). Their method consisted of data preprocessing and filtering, feature extraction, and finally classification. Similarly to us, they used *word2vec* embeddings to get vector representations of words and essays. Unlike us, they used a deep CNN for classification. To our knowledge, the results they achieved (displayed in Table 3) represent state-of-the-art performance on this dataset.

3. Essays Dataset

As we mentioned earlier, we used the *essays* dataset, which is the result of research by Pennebaker and King (1999). It consists of 2467 *stream-of-consciousness* essays written by 34 psychology students between 1993 and 1996 (Tighe et al., 2016). Each essay is accompanied with five binary labels, one for each of the Big5 personality traits. Specifically, each entry in the dataset is in the format `author_id, essay, ext, neu, agr, con, opn`, where the binary labels for traits are represented with `y` or `n`.

Trait distribution in the dataset is shown in Table 1. The values in the first row are the absolute numbers of essays with a positive label for the trait in that column, while the ratio of such essays in the dataset is given in the second row. A more detailed statistical analysis of the traits can be found in (Pennebaker and King, 1999) and (Mairesse et al., 2007).

The essays themselves come in a lot of shapes and sizes: the minimal number of words and sentences in an essay is 39 and 1, while the maximal numbers are 324 and 2964, respectively. The average number of words and sentences is 742.4 and 48.6. While a diverse dataset is generally desirable, such drastic differences in length can pose a problem, especially for models such as LSTMs. Furthermore, many examples are incomplete in some way: the essay with 39 words actually abruptly ends mid-word, while the essay with a single sentence doesn't contain any interpunction.

4. Our Approach

Our goal was to design a system that would facilitate experimentation so that we could easily try different models using different features. The models we implemented can roughly be categorized into three groups: (1) true baselines,

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 1–5, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

Table 1: Trait distribution in the *essays* dataset.

EXT	NEU	AGR	CON	OPN
1276	1233	1310	1253	1271
51.7%	49.9%	53.1%	50.79%	51.52%

(2) static models, (3) and sequence-based models. Each group has a different set of features at its disposal. All models simplify the multilabel classification task by separating it into five independent binary classification problems.

4.1. Data Preprocessing

As part of the preprocessing step, we discard the `author_id` field and convert the `y/n` labels into numerical `1/0` labels. Then we perform sentence- and word-level tokenization on lowercased essays. At this point, every example contains three views of an essay: (1) a raw essay, (2) a list of sentences in the essay, (3) a list of words in the essay. Tokenization is performed using the *punkt* tokenizer from the NLTK framework. Finally, the dataset is split into standard train/valid/test subsets with a 60/20/20 ratio, respectively.

4.2. Feature Extractors

A feature extractor is in charge of converting a dataset of essays into a dataset of fixed-size vector representations, to be used by static models. Any and all extractor parameters are initialized based on the train split of the dataset to avoid data leakage. Initialized extractors are then used to extract features from all three splits of the dataset.

To make extraction flexible and robust, the extraction method receives all three views of the essay. For example, our custom capitalization extractor requires raw essays in order to detect capitalized letters, while a *word2vec* extractor requires a list of words in the essay. All the implemented extractors are shown in Table 2.

4.3. True Baselines

True baselines consist of two rudimentary models which don't rely on any of the features, nor the essays themselves. The first baseline is a dataset-agnostic random classifier (RC), and the second one is a most common class classifier (MCC) which classifies all examples with the majority label for each of the traits, based on the distribution in the train split.

4.4. Static Models

Static models refer to classifiers whose input is a fixed-size vector representation of an essay. This group consists of three classifiers: (1) a fully connected neural network (FC), (2) a support vector machine (SVM), (3) and a naive Bayes SVM (NBSVM).

We implemented the FC model using the PyTorch framework, and the SVM implementation is taken from scikit-learn. Both of these models utilize the features generated by feature extractors. On the other hand, we used a pre-built NBSVM implementation[1] (Wang and Manning, 2012) designed to work exclusively with bag-of-words features.

4.5. Sequence-Based Models

Sequence-based models refer to models which take into account the sequential nature of essays - each essay is a sequence of words or sentences. One of the most popular models for sequential data is an LSTM cell, which we implemented using PyTorch.

We tried training LSTMs with sequences of words as well as sentences. In both cases, the elements of the sequence first had to be converted to their corresponding vector representations. Since the inputs to sequence-based models do not have a fixed dimension, we couldn't use the feature extractors as described in Section 4.2..

Instead, when working with word sequences, we used Google's 300-dimensional embeddings obtained on the Google News corpus. The embeddings were loaded and processed using the *gensim*[2] library. When working with sentence sequences, we used two different pre-trained word embeddings with a larger dimensionality (Pagliardini et al., 2018): 600-dimensional *sent2vec-wiki-unigrams* obtained on English Wikipedia and 700-dimensional *sent2vec-toronto-books* obtained on BookCorpus. The word embeddings were combined into sentence embeddings using the *epfml/sent2vec*[3] library.

4.6. Data Augmentation

It is known that LSTMs (and recurrent neural networks in general) struggle with sequences longer than a few dozen words. As stated in Section 3., an average essay contains over 700 words and around 50 sentences. This means that even the average essay is far too long to be adequately processed by an LSTM.

An additional problem for LSTMs is the great discrepancy in essay lengths. Namely, when the dataset is being batched, every instance in the batch is zero-padded to match the length of the longest instance. A big difference in length can result in some examples having more nil-vectors than actual useful information.

To address these issues, as well as the relatively small size of the dataset, we split each essay into several chunks, with each chunk having a minimum of C words (C is a hyperparameter). Splitting was implemented to only occur on the position of an interpunction symbol (`.,!?`). Each chunk was assigned the same labels as the essay from which it was taken. This resulted in a dataset of more than 40 thousand examples, the vast majority of which are similar in length.

Because the essays are already scarce with emotion, many of the examples generated by chunking were completely void of emotionally charged words. Furthermore, the fewer words there are in a chunk, the greater the chance that the dataset already contains a very similar chunk. This can lead to contradictory examples if the two chunks come from essays with different trait labels, thus making the training process even more difficult.

[1] `https://github.com/mesnilgr/nbsvm/`
[2] `https://radimrehurek.com/gensim/`
[3] `https://github.com/epfml/sent2vec`

Table 2: Feature extractors used for static models.

Extractor	Semantics	Vector dimension
Capitalization	Number of uppercase letters; normalized by sentence count	1
WordCount	Number of words; normalized using mean and SD of word counts in the train split	1
Interpunction	Number of periods, exclamation and question marks; normalized by sentence count	3
RepeatingLetters	Number of letters that were repeated 3 or more times	1
TF-IDF	TF-IDF vectors; vocabulary V built only on train set	$\|V\|$
W2V	Averaged vector representations of words in the essay	300
S2V	Averaged vector representations of sentences in the essay	600 - 700

A possible solution for the described problem was found in (Majumder et al., 2017): removing every emotionally void sentence from every essay. A sentence is considered emotionally void if it has no emotionally charged words. The emotional charge of a word is determined by comparing the word against a known set of emotionally charged words - in this case the NRC Emotion Lexicon[4] (Mohammad and Kiritchenko, 2015). We expanded on this idea and implemented two different variants of filtering: (1) removing sentences from raw essays, and then chunking the essays; (2) chunking the essays, and then removing emotionally void chunks. The second approach is motivated by the desire to remove useless and problematic examples from the generated dataset. Emotional dropping improved the performance of LSTM cells, with the second variant bringing greater benefits.

5. Results

As previously stated, we performed all model training and evaluation on the *essays* dataset. The evaluation results are shown in Table 3 and Table 4. The tables show accuracies and F1 measures of the state-of-the-art model from Majumder et al. (2017), the MCC baseline, and most of the models with which we experimented.

We ran various combinations of features and models, but displayed only the best ones. Each of the models was independently trained 10 times. All the metrics from those 10 runs were averaged and their standard deviation was calculated. It is important to note that accuracy is an acceptable performance metric on this dataset because the traits are balanced, as we have shown in Table 1. Nonetheless, we also show the achieved F1 scores and their standard deviations. To enable reproducibility we split the dataset once before all the runs, and we set up the same random number generator seed for all our experiments.

It can be seen that our NBSVM models achieve higher accuracy than the state-of-the-art on openness and neuroticism, but it should be noted that Majumder et al. (2017) evaluated their results using cross-validation, and we only split the dataset once, creating train/valid/test subsets.

6. Dataset Commentary

Human personality is very complex in its nature and determining the Big Five traits solely from text is a very challenging task. We feel that the *essays* dataset makes the problem even harder, primarily due to the stream-of-consciousness nature of the essays. Such essays exhibit the thoughts of the person writing them, but such thoughts may not reveal enough to determine the author's personality traits, as they often lack emotional expression. This is backed by the fact that dropping emotionally neutral sentences improved performance.

Furthermore, the author's thoughts are often influenced by their surroundings. If the author is not trained in controlling and structuring their thoughts for the essay, which is likely the case with most students that wrote them, there is a lot of noise in the text. For example, some essays are just describing the room the author was in at the time of writing. Because of this, noise some models have difficulty grasping the essence of the author's traits. Another downside of the noise is that the essays become unnecessarily long, which is a huge problem for models like the LSTM cell. We addressed the problem of lengthy essays by using essay chunking, previously described in Section 4.6.

In contrast to essays, posts from social media like Twitter or Facebook offer a deeper insight into people's perspectives and attitudes. The nature of social media posts is such that authors often express their opinions in an emotional manner, with emotions ranging from anger and frustration all the way to joy and excitement. Moreover, such posts are often shorter, and in the case of Twitter they even have an upper bound. This means the information present in them could be acquired faster and easier. For these reasons, we believe that social media datasets are better suited for the task of personality trait classification.

7. Conclusion

The *essays* dataset proved to be a challenging dataset for the task of personality trait classification. We managed to obtain very good results with our models, with some coming close to state-of-the-art models. In spite of that, we still believe that this dataset has shortcomings and makes the task more difficult than it could have been.

In future work, we would use cross-validation to better evaluate our models, and to be able to directly compare them to the state-of-the-art. Furthermore, we would explore the effects of emotionally charged words in essays to a greater extent. Finally, it could be informative to compare the performance of one model on several different datasets.

[4] http://saifmohammad.com/WebPages/

Table 3: Accuracies of models on each of the traits. † NBSVM used uni+bi+tri+quadgrams. ‡ NBSVM used uni+bigrams. * NBSVM used uni+bi+trigrams.

Model	OPN [$\% \pm \sigma$]	CON [$\% \pm \sigma$]	EXT [$\% \pm \sigma$]	AGR [$\% \pm \sigma$]	NEU [$\% \pm \sigma$]	AVG [$\% \pm \sigma$]
(Majumder et al., 2017)	57.30	**62.68**	58.09	**56.71**	59.38	**58.83**
NBSVM	**63.08**†	57.61†	58.01†	52.94‡	**60.45***	58.42
MCC	52.13	51.12	54.36	52.13	49.90	51.93
SVM-CUSTOM	51.32	54.77	50.10	53.55	52.54	52.45
SVM-BOW	52.13	51.12	54.36	52.13	49.90	51.93
SVM-W2V	52.13	51.12	54.36	52.13	49.90	51.93
SVM-S2V	52.13	51.12	54.36	52.13	50.51	52.05
SVM-CUSTOM,BOW,W2V	60.45	57.20	**58.42**	53.14	58.62	57.57
LSTM	51.54 ± 1.49	49.43 ± 0.45	53.08 ± 1.42	52.41 ± 0.94	51.32 ± 0.98	51.56 ± 1.06
BiLSTM	51.46 ± 1.52	49.13 ± 0.91	52.27 ± 1.66	52.03 ± 0.45	50.20 ± 1.16	51.02 ± 1.14
LSTM-CHUNK	57.93 ± 1.64	52.37 ± 2.55	51.40 ± 3.27	52.11 ± 0.06	50.20 ± 0.10	52.80 ± 1.52
LSTM-CHUNK+EMOv1	58.48 ± 2.26	52.84 ± 1.57	51.99 ± 1.88	52.09 ± 0.12	51.78 ± 3.28	53.44 ± 1.82
LSTM-CHUNK+EMOv2	59.59 ± 1.59	51.83 ± 0.85	50.97 ± 2.25	52.27 ± 0.37	59.43 ± 2.72	54.82 ± 1.56
BiLSTM-CHUNK+EMOv2	58.48 ± 2.28	51.54 ± 0.96	51.30 ± 2.50	52.11 ± 0.06	52.52 ± 1.98	53.19 ± 1.56
FC-CUSTOM	51.72 ± 1.37	51.87 ± 1.35	50.63 ± 0.93	52.37 ± 0.70	52.80 ± 1.05	51.88 ± 1.08
FC-BOW	60.93 ± 0.39	58.42 ± 0.84	54.24 ± 0.44	49.68 ± 0.45	60.00 ± 0.39	56.65 ± 0.50
FC-W2V	62.23 ± 0.56	58.26 ± 1.01	52.86 ± 1.38	51.60 ± 0.23	57.79 ± 0.33	56.55 ± 0.70
FC-S2V	60.89 ± 0.72	58.44 ± 0.60	55.46 ± 0.86	52.31 ± 0.41	57.28 ± 0.59	56.88 ± 0.64
FC-CUSTOM,BOW,W2V	62.66 ± 0.74	58.62 ± 0.40	54.58 ± 0.39	52.27 ± 0.31	59.45 ± 0.40	57.52 ± 0.45

Table 4: F1 scores of models on each of the traits. † NBSVM used uni+bi+tri+quadgrams. ‡ NBSVM used uni+bigrams. * NBSVM used uni+bi+trigrams.

Model	OPN [$\% \pm \sigma$]	CON [$\% \pm \sigma$]	EXT [$\% \pm \sigma$]	AGR [$\% \pm \sigma$]	NEU [$\% \pm \sigma$]	AVG [$\% \pm \sigma$]
(Majumder et al., 2017)	n/a	n/a	n/a	n/a	n/a	n/a
NBSVM	68.07	61.65	64.74	60.81	62.57	63.57
MCC	**68.53**	67.65	**70.43**	**68.53**	nan	**68.79**
SVM-CUSTOM	58.76	66.17	50.40	67.39	44.29	57.40
SVM-BOW	**68.53**	67.65	**70.43**	**68.53**	nan	**68.79**
SVM-W2V	**68.53**	67.65	**70.43**	**68.53**	nan	**68.79**
SVM-S2V	**68.53**	67.65	**70.43**	**68.53**	7.58	56.55
SVM-CUSTOM+BOW+W2V	62.14	58.22	61.54	57.46	59.20	59.71
LSTM	58.89 ± 11.62	62.49 ± 1.91	58.95 ± 10.51	64.23 ± 1.34	52.72 ± 7.39	59.46 ± 6.55
BiLSTM	59.14 ± 9.93	61.09 ± 1.75	55.48 ± 13.24	64.90 ± 3.62	45.98 ± 14.81	57.31 ± 8.67
LSTM+CHUNK	59.57 ± 7.11	$nan \pm nan$	53.60 ± 11.75	68.52 ± 0.05	$\mathbf{66.80 \pm 0.05}$	62.12 ± 4.74
LSTM+CHUNK+EMOv1	62.06 ± 5.86	65.80 ± 3.01	56.02 ± 6.94	68.48 ± 0.09	64.57 ± 5.37	63.39 ± 4.25
LSTM+CHUNK+EMOv2	61.91 ± 6.89	$\mathbf{67.66 \pm 0.17}$	54.87 ± 9.07	68.38 ± 0.19	59.24 ± 9.54	62.41 ± 5.17
BiLSTM-CHUNK+EMOv2	58.86 ± 12.25	66.29 ± 2.83	54.98 ± 13.55	68.52 ± 0.05	55.41 ± 21.54	60.81 ± 10.04
FC-CUSTOM	58.34 ± 1.92	60.48 ± 3.93	56.30 ± 1.96	68.01 ± 0.75	47.82 ± 7.19	58.19 ± 3.15
FC-BOW	62.40 ± 0.77	60.61 ± 1.53	58.09 ± 0.85	57.08 ± 0.61	60.38 ± 1.24	59.71 ± 1.00
FC-W2V	63.44 ± 2.27	61.56 ± 1.23	53.70 ± 8.01	57.98 ± 0.55	53.30 ± 1.63	58.00 ± 2.74
FC-S2V	62.33 ± 1.06	60.64 ± 2.13	59.67 ± 2.87	57.20 ± 1.47	57.68 ± 2.08	59.50 ± 1.92
FC-CUSTOM+BOW+W2V	63.62 ± 0.71	59.81 ± 0.50	60.15 ± 0.88	66.77 ± 0.47	57.73 ± 0.80	61.61 ± 0.67

References

Matej Gjurković, Mladen Karan, Iva Vukojević, Mihaela Bošnjak, and Jan Šnajder. 2020. Pandora talks: Personality and demographics on reddit.

Francois Mairesse, Marilyn Walker, Matthias Mehl, and Roger Moore. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *J. Artif. Intell. Res. (JAIR)*, 30:457–500, 09.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, and Erik Cambria. 2017. Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, 32:74–79, 03.

Saif M Mohammad and Svetlana Kiritchenko. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, 31(2):301–326.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 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)*, pages 528–540, New Orleans, Louisiana, June. Association for Computational Linguistics.

Gregory Park, H. Schwartz, Johannes Eichstaedt, Margaret Kern, Michal Kosinski, David Stillwell, Lyle Ungar, and Martin Seligman. 2014. Automatic personality assessment through social media language. *Journal of personality and social psychology*, 108, 11.

James Pennebaker and Laura King. 1999. Linguistic styles: Language use as an individual difference. *Journal of personality and social psychology*, 77:1296–312, 01.

Daniele Pizzolli and Carlo Strapparava. 2019. Personality traits recognition in literary texts. In *Proceedings of the Second Workshop on Storytelling*, pages 107–111, Florence, Italy, August. Association for Computational Linguistics.

Edward P. Tighe, Jennifer C. Ureta, Bernard Andrei L. Pollo, Charibeth K. Cheng, and R. D. Bulos. 2016. Personality trait classification of essays with the application of feature reduction. In *SAAIP@IJCAI*.

Sida Wang and Christopher Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94, Jeju Island, Korea, July. Association for Computational Linguistics.

Together Against Hate: Different Approaches to Offensive Language Detection

Sara Bakić, Josipa Lipovac, Antonijo Marijić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{sara.bakic, josipa.lipovac, antonijo.marijic}@fer.hr

Abstract

Offensive language is omnipresent in social media. Individuals frequently take advantage of the perceived anonymity of computer-mediated communication, using this to engage in behavior many of them would not consider in real life. Cyber-violence is present in different cultures and different languages. This work explores different monolingual and multilingual fine-tuning settings and how they affect the offensive language detection ability of an attention-based multilingual model, XLM-RoBERTa (XLM-R). Offensive language detection task was realized in five languages: Arabic, Danish, English, Greek and Turkish. Experimenting with four different setups, we explore if a single model for all languages can achieve results comparable to monolingually fine-tuned models, search for benefits of having a high-resource mediator language in fine-tuning process and explore if the cross-lingual ability of XLM-R can overcome differences in script.

1. Introduction

The majority of the work in natural language processing (NLP) is focused on the English language. Datasets for various tasks in English are the biggest, especially ones with labeled data. Although models specialized for a certain non-English language have been made, research has gone in the direction of making one architecture for many languages. Also, low-resource languages benefit from the cross-lingual transfer to a great extent. As a result, models with transformer-based structure have been created and trained on an immense amount of data in over 100 languages (Devlin et al., 2019; Conneau et al., 2020).

These pretrained models need to be fine-tuned on the dataset of the downstream task. Although the model is already pretrained multilingually, fine-tuning on more than one language for a specific task can also be beneficial. Some researchers experimented with different combinations of fine-tuning (Tang et al., 2020; Kondratyuk, 2019), but to the best of our knowledge, systematic analysis of a various fine-tuning configurations hasn't been done.

In this paper we present the comparison between 4 different configurations of fine-tuning and test three hypotheses regarding the cross-lingual transfer and its dependency on the alphabet of the language and size of the dataset. The fine-tuning is done on the OffensEval-2020 (Zampieri et al., 2020) multilingual dataset. The task is to detect the offensive language in tweets. Data is provided in 5 languages: English (Zampieri et al., 2019), Danish (Derczynski and Sigurbergsson, 2020), Greek (Pitenis et al., 2020), Turkish (Çöltekin, 2020) and Arabic (Mubarak et al., 2020). We used pretrained XLM-RoBERTa (XLM-R) model (Conneau et al., 2020).

In the following section we provide an overview of the related work on a multilingual fine-tuning of the transformer-based architectures. Next, we present the database structure and the preprocessing. Then we describe the model used in experiments and optimization of the hyperparameters. In the next section we present experiments and the hypotheses. We conclude the paper by presenting and discussing our results.

2. Related Work

Recent works explored multilingual models for numerous tasks in the NLP. Some of the newest models are multilingual BERT (Devlin et al., 2019), mBART (Liu et al., 2020) and XLM-R. All three models have a transformer-based architecture which uses an attention mechanism and they are pretrained on a vast amount of data in many languages. Pretrained models are fine-tuned on the training data for a specific downstream task. Fine-tuning can be monolingual or multilingual where transfer of knowledge between languages can be beneficial.

Tang et al. (2020) proposed a new approach for multilingual translation models that have multilingual pretraining followed by multilingual fine-tuning. Previous work used bilingual fine-tuning (Liu et al., 2020) which does not use the full potential of the multilingual pretraining. With this new approach models can translate many languages to many other languages. They presented three different configurations of the fine-tuning: *many-to-one*, *one-to-many* and *many-to-many* languages. The authors compared their model with three baselines: a model with bilingual training built from scratch, model with bilingual fine-tuning and multilingual model trained from scratch. They got better results in the *many-to-one* setting, but worse results on the *one-to-many* and *many-to-many* than the baselines.

Kondratyuk (2019) presented a system that uses fine-tuning a multilingual BERT model (Devlin et al., 2019) for lemmatization and morphology tagging. They showed that the model can be improved significantly by incorporating multilingual pretraining which allows the model to incorporate cross-lingual information useful for morphological parsing. The authors used three separate configurations: *mono*, *multi* and *multi+mono*. In the *mono* configuration, they fine-tuned the model on each treebank separately. In the *multi* configuration, they fine-tuned the model as in *mono*, but the data from all languages is concatenated together. Finally, in the *multi+mono* configuration there is two-stage process: fine-tuning the model as in *multi* and then fine-tuning the model monolingually as in *mono*. Their results show that two-stage *multi+mono* fine-tuning

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 6–10, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

can provide significantly better results than the *mono* configuration for almost every treebank.

In line with previous work, we explore the benefits of multilingual fine-tuning on the downstream task and cross-lingual information transfer. The goal of our work is not to get state-of-the-art results on the downstream task, but to systematically investigate different multilingual fine-tuning configurations. While Kondratyuk (2019) has *mono*, *multi* and *multi+mono* setups, we added configurations where English is a mediator, languages are grouped by families and language is in two different alphabets.

3. Dataset

3.1. OffensEval Dataset

Dataset used in this work was taken from OffensEval[1] (Zampieri et al., 2020) competition site. It is a multilingual dataset available in 5 target languages (English, Arabic, Danish, Greek and Turkish). Although there is the latest English dataset on the competition page - SOLID[2] dataset, we decided to use an older version - OLID[3] dataset. The SOLID dataset consists of over 9,000,000 annotated tweets, while OLID contains only around 14,000 of them. We chose OLID to get a balance between datasets of different languages.

Annotation schema proposes hierarchical modeling of offensive language. It classifies each example using the following three-level hierarchy:

- Level A - Offensive language detection
- Level B - Categorization of offensive language
- Level C - Offensive language target identification

Since our main goal was to show the influence of the multilingual component on the model for detecting offensive language, dataset level A was used. In Table 1 the distribution of tweets is shown. *OFF* label represents text containing inappropriate language, insults, or threats. *NOT* label represents text that is neither offensive, nor profane. For validation purposes, the train set is shuffled and 10% of the data is separated.

Table 1: Distribution of tweets by languages and labels

	Arabic		Danish		English		Greek		Turkish	
	OFF	NOT	OFF	NOT	OFF	NOT	OFF	NOT	OFF	NOT
Train	1589	6411	387	2588	4400	8840	2486	6257	6143	25627
Test	402	1598	41	289	240	620	242	1302	716	2812
Total	1991	8009	428	2877	4640	9460	2728	7559	6859	28439

3.2. Preprocessing

For preprocessing, tweet-preprocessor[4] library was used and it included various text transformations. Cleaning, tokenizing and parsing of the following items was performed: URLs, hashtags, mentions, reserved words (RT, FAV), emojis and smileys.

Arabic and Greek dataset are written in the original script, i.e. in Arabic and Greek alphabets. From these datasets, additional ones were created in such a way that the original datasets were converted to the Latin alphabet. Preprocessing of datasets of the Arabic and Greek alphabets was not possible using the tweet-preprocessor library due to the fact that the mentioned library functions only with the Latin alphabet. Therefore, preprocessing was performed on previously created datasets in the Latin alphabet. After preprocessing these datasets, the obtained results were subsequently converted to the original alphabets. It was ensured that the tags obtained by preprocessing (@USER, $HASHTAG$, ...) remain in their original form.

4. Model and Hyperparameter Optimization

We did all of our experiments on XLM-R base model. A pretrained model was obtained from HuggingFace (Wolf et al., 2020) and was adjusted to the offensive-language classification task. The pretrained model consisted of a transformer encoder which outputted token-level representations for the sequence. To perform a classification task, a classification head was added on top of the XLM-R encoder with two-dimensional output denoting whether the tweet is offensive or not. Having a model with architecture suited for offensive language classification, we started the fine-tuning process using the aforementioned datasets. An overview of the architecture of the model we used for this work is illustrated in Figure 1.

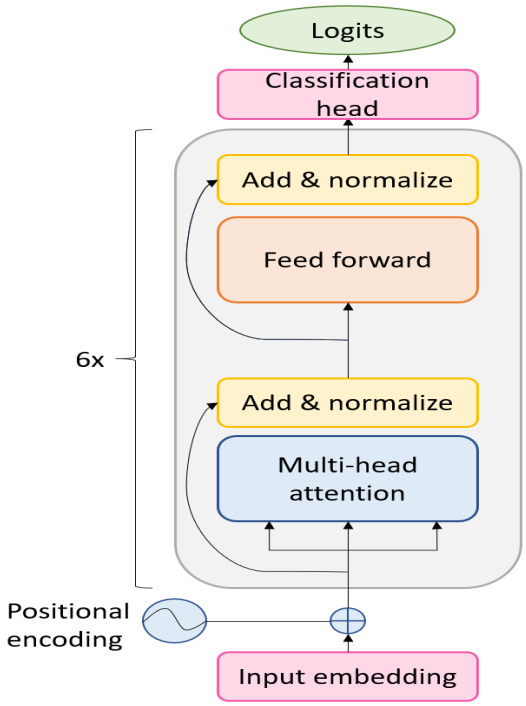


Figure 1: Scheme of the model - XLM-R + classification head

Since the embedding model was pretrained, a lot of hyperparameters regarding the model architecture are fixed

[1] https://sites.google.com/site/offensevalsharedtask/home
[2] Semi-Supervised Offensive Language Identification Dataset
[3] Offensive Language Identification Dataset
[4] https://pypi.org/project/tweet-preprocessor/

(e.g. number of embedding layers, dimensions of representation vector, number of attention heads, ...) but there are still some hyperparameters that can have a significant impact on the classification results. These hyperparameters are mostly related to optimization. We experimented with different learning rates (10^{-4}, 10^{-5}, 10^{-6}), optimizers (SGD, Adam, AdamW) and schedulers (linear scheduler, linear scheduler with warmup steps). We also explored the influence of setting maximum sequence length to different values and tested the model with maximum sequence length set to 64 and 128 tokens. Since tweets usually have a length constraint of $\sim$300 characters, these values are appropriate for experimenting with tweets. The best hyperparameters were chosen through a grid search of the hyperparameters listed above.

It should be noted that the number of fine-tuning epochs was not set as hyperparameter but we rather performed early-stopping regularization by stopping the training if the F1 score on the validation set was not improved for two consecutive epochs.

5. Experimental Setup

Our work is based on 4 different experimental setups:

1. Model fine-tuning on all languages at once
2. Model fine-tuning on each language separately
3. Model fine-tuning firstly on English and then on each language separately
4. Model fine-tuning on Indo-European language family with two different approaches:
   (a) fine-tuning on all languages at once with Greek in Greek alphabet
   (b) fine-tuning on all languages at once with Greek converted to Latin alphabet

Table 2: Language proximity [5]

	Arabic	Danish	Greek	Turkish
English	83.6	**20.6**	69.9	92.0
Arabic	-	85.3	93.5	97.2
Danish	-	-	69.7	95.5
Greek	-	-	-	93.6
Turkish	-	-	-	-

In Table 2 the values of language proximity are presented. It can be concluded that languages belonging to the same language family have a lower value of language proximity. Examples are the languages of the Indo-European language family: English, Danish and Greek.

In order to show whether there is an impact on the results of offensive language detection when using multilingualism, several hypotheses have been set:

- **Hypothesis 1**: It is possible to achieve equally good results when training one model for all languages simultaneously compared to training a model for each language separately
- **Hypothesis 2**: Training together with other languages, and especially with English as a mediator, does not contribute to improving the results in the Danish dataset
  - The dataset for the Danish language is quite small. Since they belong to the same family, Danish and English have a low value of language proximity (Table 2).
- **Hypothesis 3**: The language script does not affect the cross-lingual component of XLM-R
  - Experimental setup 4 involves training with a Greek dataset composed of tweets in the Greek alphabet (4.a) and tweets in the Latin alphabet along with English and Danish dataset (4.b). We decided to explore the influence of language script in a multilingual setting in order to keep the positive cross-lingual influence of languages that are relatively close to Greek (the remaining two Indo-European languages in this case) therefore increasing the model's ability to overcome the script barrier.

6. Results and Discussion

Table 3: Average F1 scores per experimental setup

	1st	2nd	3rd	4th a)	4th b)
English	**0.69**	**0.69**	-	0.66	**0.69**
Danish	0.53	0.12	**0.60**	0.51	0.50
Greek	0.62	0.66	0.66	**0.67**[6]	0.66[7]
Arabic	0.59	0.60	**0.62**	-	-
Turkish	0.63	0.63	**0.65**	-	-

We trained the models for each of the tasks 5 times with the best hyperparameters. While AdamW and linear scheduler have shown as the best choice for all of our experiments, different learning rate and the maximum sequence length yielded the best results across different experiments. Most of the experimental setups achieved the best results with learning rate set to 10^{-5} and the maximum sequence length set to 128. The exceptions are Danish in setup 2 which achieved the best results with a learning rate set to 10^{-6} and maximum sequence length set to 128 and Arabic in setup 3 which achieved the best results with a learning rate set to 10^{-5} and maximum sequence length set to 64. The average F1 results are shown in the Table 3. The results for English datasets are the same for setup 1, setup 2 and setup 4 b). The result for setup 4 a) is slightly lower. The best result obtained for the Danish dataset is one obtained on setup 3. We can notice from the results that the lowest F1 scores are for the Danish dataset. This can be justified by the fact that Danish dataset is very small and

[4] http://www.elinguistics.net/Compare_Languages.aspx
[5] http://www.elinguistics.net/Compare_Languages.aspx
[6] In greek alphabet
[7] In latin alphabet

Table 4: Statistical tests

	H0	H1	p-value	Result Discussion
Hypothesis 1	$\mu 1 = \mu 2$ Multi and monolingual setting achieve the same results	$\mu 1 \neq \mu 2$ Multilingual results are different from monolingual results	0.45	With α = 0.05 we didn't manage to dispute H0
Hypothesis 2-1	$\mu 1 = \mu 2$ Multi and monolingual setting achieve the same results on Danish	$\mu 1 < \mu 2$ Multilingual results are better than monolingual on Danish	0.0004	With α = 0.05 we managed to dispute H0
Hypothesis 2-2	$\mu 1 = \mu 2$ Monolingual setting achieves equal results as having English as mediator on Danish	$\mu 1 < \mu 2$ Having English as mediator improves results on Danish	0.0002	With α = 0.05 we managed to dispute H0
Hypothesis 3	$\mu 1 = \mu 2$ Results on Greek are equal despite different scripts	$\mu 1 \neq \mu 2$ Results on Greek depend on script of the tweets	0.74	With α = 0.05 we didn't manage to dispute H0

unbalanced. The best result obtained for the Greek dataset is one on setup 4 a). In that case Greek dataset contained tweets on the Greek alphabet. The best results obtained for Arabic and Turkish datasets are on setup 3.

We performed statistical processing on the obtained data, relying on the hypotheses set in the previous section. All descriptions of processed statistical tests are shown in Table 4. In the case of Hypothesis 1 we didn't manage to dispute H0 by a two-tailed t-test. This shows that there is no statistically significant difference between the multi and monolingual results. We cannot say statistically correctly that our results do confirm our Hypothesis 1, but they definitely do not refute it. In the case of Hypothesis 2, it was first necessary to check whether there was a significant difference in the results obtained for task 1, task 2 and task 3. Differences in mean values were checked using ANOVA. Obtained p-value was $3 \cdot 10^{-8}$. After that two different t-tests are performed (Table 4, Hypothesis 2-1 and 2-2). As can be seen from the table, the results of one-tailed t-tests show that our Hypothesis 2 is not valid. Training together with other languages, and especially with English as a mediator, improves the results in the Danish dataset. In the case of Hypothesis 3, we failed to dispute H0 by a two-tailed t-test. This shows that there is no statistically significant difference between the results obtained after training on the Greek alphabet and training on the Latin alphabet. As in the case of Hypothesis 1, we cannot statistically correctly say that our Hypothesis 3 is valid, but the results do not refute it.

7. Conclusion

In this work, we explored different fine-tuning techniques for offensive language detection. We showed that multilingual setting for offensive language detection task achieves results comparable to a monolingual setting. Also, we showed that introducing a high-resource language, such as English, as a mediator language can be beneficial for other low-resource languages, especially for those that are more related to English, such as Danish. XLM-RoBERTa has shown as a good choice for multilingual problems for its strong cross-lingual component. Having comparable results on the Greek dataset in both Greek and Latin alphabet supports this statement.

Future work would include performing these experiments on a larger and more diverse set of languages, as well as a larger amount of data per language. That way, the overall results would be more reliable and the benefits of different fine-tuning approaches would be more noticeable.

References

Çağrı Çöltekin. 2020. A corpus of turkish offensive language on social media. ELRA.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July. Association for Computational Linguistics.

Leon Derczynski and Gudbjartur Sigurbergsson. 2020. DKhate: Danish Hate Speech Abusive Language data. 5.

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.

Dan Kondratyuk. 2019. Cross-lingual lemmatization and morphology tagging with two-stage multilingual BERT fine-tuning. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 12–18, Florence, Italy, August. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training

for neural machine translation. *Transactions of the Association for Computational Linguistics*, pages 726–742, November.

Hamdy Mubarak, Ammar Rashed, Kareem Darwish, Younes Samih, and Ahmed Abdelali. 2020. Arabic offensive language on twitter: Analysis and experiments. *arXiv preprint arXiv:2004.02192*.

Zeses Pitenis, Marcos Zampieri, and Tharindu Ranasinghe. 2020. Offensive language identification in greek.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020). In *Proceedings of SemEval*.

Is Context Enough? Combining BERT and Domain Knowledge for Bullying Traces Detection

Ana Barić, Martin Čolja, Ana Leventić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{ana.baric, martin.colja, ana.leventic}@fer.hr

Abstract

Bullying detection keeps being relevant in the age of social media. We can leverage high information exchange in social media sites to detect bullying events early. We focus on Twitter as the main source of information. While most of the previous work accepts the limitations of the data, we try to expand our model beyond keyword specific dataset by relying on the tweet context. We combine BERT with a convolutional neural network to produce tweet embedding, which we combine with domain-specific knowledge in the form of hand-crafted features.

1. Introduction

It's undeniable that bullying is present in our society. Bullying can happen anywhere and affects everyone. Being bullied in childhood is connected to depression and mood disorders (Bauman et al., 2013), while any involvement in the bullying process may lead to the increase of future emotional trauma (Wolke et al., 2014).

It is not surprising that there is an effort to prevent bullying and educate about its consequences and harmful effect on society. Bullying detection is the first step of preventing bullying. In the age of social media, information is easily accessible and can be used to identify bullying traces. Detecting the bullying traces before they are posted on social media, could make it possible to redirect the author to pages dedicated to informing about bullying and supporting those in need.

In this paper, we propose using BERT (Bidirectional Encoder Representations from Transformers) embeddings for the task of bullying traces classification (Devlin et al., 2019). The motivation behind it lies in its characteristic to learn the context of a word based on its neighbouring words. Our goal is to inspect how using contextualised vector representation performs on the task. Moreover, we assume that using domain knowledge in the form of additional features could improve the performance on the given task. Naturally, we experiment and evaluate the combination of hand-crafted features with a strong attention model.

In Section 2. we explain similar work followed by describing the dataset in Section 3. We will explain our setup in Section 4., while discussing results in Section 5. The paper concludes with an open discussion in Section 6.

2. Related work

The popularity increase of social media has spiraled a lot of interest around the task of bullying trace detection. Xu et al. (2012) presented baseline results for bullying trace classification with feature representations that included unigrams, bigrams, and POS-tagging. Di Capua et al. (2016) proposed an unsupervised approach to detect cyberbullying in social media with hand-crafted features that capture semantic and syntactic behavior in bullying traces. Lim and Madabushi (2020) used an ensemble of BERT and corpus-level information in form of TF-IDF to detect the offensive language in social media.

In this paper, we propose a deep-learning approach for bullying trace detection that incorporates contextual knowledge from the pre-trained BERT model and explore the behavior of the proposed system when it is combined with domain knowledge.

3. Dataset

This dataset is a part of the research on understanding and fighting bullying with machine learning (Xu et al., 2012). The dataset contains 7,321 tweets in English as well as in other languages. Our research focused only on tweets written in English since we use advanced pre-trained models with this requirement. Skipping the tweets deleted by Twitter and tweets which are not written in English leaves us with 2,528 annotated tweets. The creators of this enriched dataset included tweets solely based on keywords such as bully, bullied, etc. This collection method is limiting since it introduces a divide between the collected data and day-to-day tweets. It enables the models to overadjust to the dataset and underperform in real-world applications. The advantage of collecting data in this manner, however, is a very well-balanced dataset. A total of 1057 tweets, out of 2,528, are labeled as containing bullying traces.

4. Experimental setup

To combat the restrictions of the enriched dataset mentioned in Section 3., we wanted to rely on a model which could consider the context of the tweet. That way, the downsides of keyword-filtered data would be somewhat mitigated, while the benefit of working with a balanced dataset would remain. We obtained the tweet context by taking advantage of the bi-directional attention provided by BERT to create contextualized representations for each tweet. The next step is passing the embedding matrix into a convolutional layer. CNN provides additional n-gram information to the model (Jacovi et al., 2020), the significance of n-grams was shown by Xu et al. (2012). The CNN also flattens the embedding matrix into an embedding vector,

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 11–14, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

giving us the means of providing domain-specific knowledge in the form of hand-crafted features. The final concatenated feature vector gets passed to the fully connected layer, which performs classification.

4.1. Preprocessing

Social media posts tend to be short and often contain slang words, emoticons, hashtags, and other special tokens. Because of that, it is important to preprocess the data. Additionally, it is possible to use those special tokens as features. We replaced user mentions accompanied with "@" with "@user". Likewise, URLs were replaced with "httplink". Hashtags were not split but were treated as a single token without the symbol "#" (e.g. #bullyingIScool would be simply treated as bullyingIScool). Additionally, other special tokens such as emoticons, punctuation, and upper case words are described in Subsection 4.3. as they were used as additional features.

4.2. Embeddings

We provided context to our model by extracting only the word-piece embeddings from the pre-trained BERT-base (Devlin et al., 2019) and BERTweet model (Nguyen et al., 2020). Preprocessing steps included adding special tokens, padding to the maximum length of 60 word-piece tokens per tweet, and generating an attention mask. Each tweet was transformed to a word embeddings matrix in which each word-piece token was represented by a 768-dimensional vector. We used sum-last-four as a layered strategy for summarizing embeddings matrices of each hidden layer. Different layer strategies such as taking the eleventh hidden layer and summarizing outputs of all hidden layers were also tested, but sum-last-four gave the best results for this dataset.

4.3. Hand-crafted features

We introduce hand-crafted features into the model as means of providing domain knowledge. We standardized our hand-crafted features by removing the mean and scaling them to unit variance. Standard Scaler from SciKit-learn was used for this task (Pedregosa et al., 2011). We express a measure of negativity for each tweet by using a dictionary of words with negative undertones, from now on referred to as bad words. The dictionary can be obtained with NLTK by using Opinion Lexicon Corpus Reader (Hu and Liu, 2004). This value is represented as the frequency of the bad words in a tweet. We experimented with alternative representations, such as counting or calculating a TF-IDF score for all bad words but were unsatisfied with the results shown in Figure 1. We use the same method to calculate the frequency of punctuation symbols and emoticons. The final feature we found relevant was the use of capitalized words in a tweet. They are often used outside the rules of language to provide additional meaning. We represent this with a binary variable since it produced better results than more elaborate alternatives as seen in Figure 1. The features such as capitalization presence and frequency were never combined due to high Pearson correlation. We chose the aforementioned features based on the work of (Di Capua et al., 2016), where they achieved reasonable performance. The features varying means for positive and negative classes in Figure 1 further support this decision.

Figure 1: Mean of hand-crafted features values for positive and the negative class

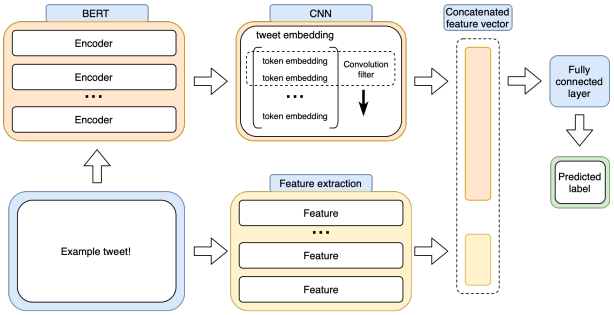


Figure 2: Combined features model components. For given tweet, BERT embeddings and hand-crafted features are extracted. BERT embeddings are sent to CNN and output is concatenated with features. Concatenated feature vector is sent to fully connected network.

4.4. Methods

In this section model architectures mainly used for the task are described.

Baseline models. For comparison, we used two baseline models. We decided to use the tweets TF-IDF score as a feature vector and an input to the model. The models themselves were SVM with the linear kernel and hyperparameter C = 1, and RandomForest of depth 20, implementation provided by the SciKit-learn (Pedregosa et al., 2011) package.

CNN classifier for BERT embeddings. The model consists of convolutional layers with multiple-sized filters. The convolution is repeatedly applied to embedded BERT tokens, followed by activation function and pooling. Due to CNN's ability to detect patterns, when applied to BERT

Table 1: Classification accuracy and f-1 scores on the test dataset. Symbol † means the model performs significantly better with $p < 0.05$ in contrast to the random forest baseline model. Symbol ‡ means the model performs significantly better with $p < 0.05$ in contrast to the both baseline models.

Model	Accuracy	F-1 score
Random Forest	0.750394	0.737776
SVM	0.767404	0.762993
BERTweet + CNN	0.765012	0.757070
BERT-Base + CNN	0.792712 ‡	0.788153 ‡
Combined (BERTweet)	0.770563 †	0.762142 †
Combined (BERT-Base)	**0.799823** ‡	**0.793516** ‡

embeddings it serves for detecting phrases and expressions. (Jacovi et al., 2020) showed that filters capture semantic ngram classes. Additionally when using max pooling, the model can separate important ngrams from the rest. The outputs from convolutional layers are concatenated to form a single feature vector which is used further in fully connected layers.

Combined features model. The model is an upgrade of the aforesaid CNN classifier. The Figure 2 shows the model components. In addition to BERT embeddings, it also extracts hand-crafted features as additional information. As in the previous model, BERT embeddings are sent through a convolutional network, but rather than forwarding them to fully connected layers, they are concatenated with additional features. The concatenated vector is then used as a new input vector to the feed-forward neural network that finally classifies the input.

ReLu was used as an activation function for both models, except for the last layer where a sigmoid function was used for classification. Furthermore, dropout is used as a regulation method. The loss calculated was binary cross-entropy loss. For training, Adam optimizer was used (Kingma and Ba, 2017).

5. Results

For the binary classification task of detecting whether a tweet represents a bullying trace, we trained two baseline models, SVM and Random Forest, and two deep models, CNN classifier for BERT embeddings and combined features model. Furthermore, for deep models, we trained two versions of the BERT model, BERT-Base that was pre-trained on Wikipedia corpus, and BERTweet that pre-trained on English Tweets.

We performed stratified 5-fold cross-validation due to the limited dataset size. For evaluation metrics, we observed both accuracy and F1-score. Mean-summary test results for all models are presented in Table 1. To compare the models, we used a one-sided paired t-test with the null hypothesis that there is no difference in performance between two models with the significance level $\alpha = 0.05$.

We rejected the null hypothesis in cases in which we compared deep learning BERT-Based model's performance with the baseline performance. When comparing the performance of Combined (BERT-Base) with BERT-Base+CNN model, we could not reject the null hypothesis for either accuracy($p = 0.24$) or F1-score($p = 0.28$). Performance comparison between Combined (BERTweet) and BERTweet + CNN yielded the same result of not having enough evidence to reject the null hypothesis for both accuracy($p = 0.32$) and F1-score($p = 0.36$). Lastly, from the comparison of the BERT-Base and BERTweet model, we can conclude that the performance difference is statistically significant, in favor of the BERT-Base model. Additionally, comparing Combined (BERT-Base) and Combined (BERTweet) produced analogous results, meaning Combined (BERT-Base) performed better than Combined (BERTweet) model.

The obtained results suggest that our hand-crafted features did not significantly improve the model's performance and that standalone contextualized features can outperform baseline models. Our assumption that explains the lack of statistical improvement with the hand-crafted features is that the weight contribution of each hand-crafted feature did not have a strong enough impact when combined with contexualized BERT features in the fully connected layer.

6. Conclusion

In this paper, we attempt to take into consideration the context of a tweet when attempting to recognize bullying traces. In doing so we also include hand-crafted features to see if they improve model performance or if relying on context alone is enough. Our results showed the efficiency of BERT attention mechanism by outperforming the baseline models. The hand-crafted features we extracted insignificantly improved model performance. The model can be further improved by fine-tuning BERT to the used dataset. For future work, we propose experimenting with additional features and better fine-tuning.

References

Sheri Bauman, Russell B. Toomey, and Jenny L. Walker. 2013. Associations among bullying, cyberbullying, and suicide in high school students. *Journal of Adolescence*, 36(2):341–350.

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.

Michele Di Capua, Emanuel Di Nardo, and Alfredo Petrosino. 2016. Unsupervised cyber bullying detection in social networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 432–437.

Minqing Hu and Bing Liu. 2004. Opinion lexicon information.

Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2020. Understanding convolutional neural networks for text classification.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Wah Meng Lim and Harish Tayyar Madabushi. 2020. Uob at semeval-2020 task 12: Boosting bert with corpus level information.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online, October. Association for Computational Linguistics.

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.

D. Wolke, S. T. Lereya, H. L. Fisher, G. Lewis, and S. Zammit. 2014. Bullying in elementary school and psychotic experiences at 18 years: a longitudinal, population-based cohort study. *Psychological Medicine*, 44(10):2199–2211.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 656–666, Montréal, Canada, June. Association for Computational Linguistics.

Hey, that's my line! — Speaker Identification and Catchphrase Resolution in Multiparty Dialogue

Katarina Boras, Ivana Cvitanović, Daria Vanesa Cvitković

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{katarina.boras, ivana.cvitanovic2, daria-vanesa.cvitkovic}@fer.hr

Abstract

This paper describes a solution to the speaker identification and catchphrase resolution problem in multiparty dialogue. To tackle speaker identification, we construct a support vector machine classifier and examine its behavior in regards to data from the TV show *Friends*. To resolve catchphrases, we create another dataset using paraphrase mining and test it on the constructed SVM. We hypothesize about the perceived and real impact of catchphrases and conclude that to a classifier, catchphrases aren't any more impactful than regular sentences.

1. Introduction

Both speaker identification and catchphrase resolution are multi-class classification tasks. The goal of speaker identification is to determine, from a list of characters, which one is most likely to utter the given sentence. The same procedure is done in catchphrase resolution, where a speaker must be assigned to a catchphrase. However, the issue with catchphrase resolution becomes extracting the actual catchphrases.

In the case of *Friends*, there are six main characters to map to given sentences. The goal of our system is to be able to not only identify the speaker of a line, but also to be able to connect a certain catchphrase to its owner. For example, if we feed the classifier the sentence "How you doin'?", it should map it to Joey.

The paper also studies the impact of catchphrases. Viewers of the show can certainly map a catchphrase to a character, but the question is whether they're frequent and important enough for a trained classifier to do it.

2. Related Work

The majority of previous work regarding speaker identification depends heavily on acoustic features.

Hazen et al. (2003) present two different approaches to speaker recognition and discuss their strengths and weaknesses. The first one is a text-dependent Gaussian mixture model approach, and the second one is a speaker adaptive automatic speech recognition model. These two models yield good results. Additionally, the authors fused the two stated approaches, which provided them with additional performance improvements.

While many approaches use acoustic features, there are many instances when those features are not present, and for that reason, there is a need to develop a model that works with only textual features.

Mairesse et al. (2007) claim that it is possible to recognize personality from linguistic cues by exploiting textual features from the text. They examine different feature sets and apply regression and ranking models to model the personality of the speaker.

Campbell et al. (2006) exploit the power of support vector machines to classify patterns. By using the kernel that compares sequences of feature vectors to measure similarity, they manage to distinguish the boundary between a speaker and a set of imposters. This approach differs from other models that usually produce the probability distributions of the speaker and others.

Ma et al. (2017) present a model that uses multi-document convolutional neural network (CNN) for speaker identification of characters from the TV show *Friends*. It uses sequential information by concatenating each utterance with the previous two and one subsequent utterance after the global max-pooling layer. Additionally, to boost the model's ability to identify the character, it can optionally take into account only the speaker that is appearing in the scene.

2.1. Dataset

As our main dataset, we used a dataset provided by Shilpi Bhattacharyya[1]. It contains over 90 thousand lines uttered by the 6 main characters of the show throughout all ten seasons. The distribution of lines over the characters is shown in table 1 (column E).

We split the set into train and test sets with a rather big ratio, lead by the thought that since sentences are fairly different from each other, it is best to have as much data as possible in the train set in order to get better results.

The second dataset, the one we used for catchphrase resolution, was created as follows. First, for each character we came up with a certain catchphrase that they are known for. For some characters, it was fairly easy (e.g. Monica's "I know!"), but some characters either didn't have one-line catchphrases or had ones that change throughout the show. Examples are Phoebe, who doesn't have a specific catchphrase[2], and Chandler, whose "catchphrase" is emphasizing the verb *to be* in his jokes. Catchphrases used are as follows:

[1] https://github.com/shilpibhattacharyya

[2] One could argue that "Oh, no." is somewhat of a catchphrase for her, and we did use it, but it's certainly not on the same level as other characters' catchphrases.

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 15–18, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

Table 1: Datasets in regards to characters. E – number of examples in the main train and test sets, C – number of examples in catchphrases set mined from the main train and test set, V – variance in cosine similarity between catchphrases.

Index	Name	E	C	V
0	Rachel	16736, 180	100, 14	0.0053
1	Ross	16346, 169	44, 21	0.0077
2	Monica	15098, 148	81, 14	0.0030
3	Chandler	15067, 143	100, 8	0.0014
4	Joey	14936, 157	100, 7	0.0115
5	Phoebe	13330, 144	53, 13	0.0081

- "I know!" for Monica,
- "Hi." and "We were on a break!" for Ross,
- "Nooo!" for Rachel,
- "Oh, no." for Phoebe,
- "Could I *be* any funnier?" for Chandler, and
- "How you doin'?" for Joey.

Since the characters utter their catchphrases in a variety of situations, we employed paraphrase mining to construct this dataset.

2.1.1. Paraphrase Mining

We utilize the SBERT[3] sentence transformer to find paraphrases of the designated catchphrase(s) for each character. This paraphrase mining function compares all the sentences in the main corpus against each other and returns a list of pairs and their similarity scores calculated using cosine similarity. We set up the experiment so that we extract only the similarities of preselected catchphrases with all other phrases in the corpus, for each of the characters. We also experimentally extract only those paraphrases that have a similarity score greater than 0.5. Finally, we calculate the variance of the similarity scores for each character, shown in table 1, column V. The variance is calculated over the main train set for fairer results.

Distribution over the characters for this set is shown in table 1 (column C) for train and test sets respectively. Nature of catchphrases doesn't allow the train and test sets to be very large.

2.2. System Description

Our system consists of two parts: speaker identification and catchphrase resolution. For speaker identification, we trained a support vector machine (SVM) on our main train set. For catchphrase resolution, we used the dataset created by paraphrase mining to test our trained SVM on.

2.2.1. Preprocessing

As the main dataset examples come in the format (sentence, speaker), some preprocessing needed to be done. For tokenization, we simply split the sentences into words and used words as tokens. We numericalized the tokens using Podium[4] and its vocabulary. For feature extraction, we used Podium's TF-IDF vectorizer. It transformed our collection of documents into a matrix of TF-IDF features, as well as converting all characters to lowercase.

2.2.2. Speaker Identification

For speaker identification, we used a support vector machine (SVM), implemented in the *scikit-learn*[5] library, to assign one of 6 main characters to each sentence. We trained several different models, as is explained later in section 3. and consequently shown in table 2. We conclude that the best model we can hope for is an SVM with a polynomial kernel and hyperparameters C and γ set to 5 and *scale* respectively.

2.2.3. Catchphrase Resolution

To conduct catchphrase resolution, we used the set created with paraphrase mining as described in section 2.1.1. and our trained SVM model. We tested the performance of the model on the catchphrases mined from train and test sets respectively, and obtained results shown in table 2.

3. Evaluation

We used two baselines to compare our models with. First baseline ("random") assigned a random character to each sentence, and the second one ("loudmouth") searched for the most frequent speaker and assigned them to every sentence. Their results are shown in table 2. Our best model is considerably better than both baselines.

Cross-validation was done with respect to three parameters: the kernel of the SVM, C for all models, and γ for all models except linear. Due to time and resource constraints, we weren't able to conduct an exhaustive grid search with more hyperparameter values, although we recognize that it would have been the best approach here.

For gamma values, we used *scikit-learn*'s *scale* and *auto* values that adjust to the dataset. They are calculated as shown in equations 1 and 2.

$$\gamma_{\text{auto}} = \frac{1}{n_{\text{features}}} \approx 3.058 \cdot 10^{-5} \tag{1}$$

$$\gamma_{\text{scale}} = \frac{1}{n_{\text{features}} * \sigma^2(X)} \approx 1.0166 \tag{2}$$

Confusion matrices are shown in figures 1, 2, and 3.

We display results for both the catchphrases train and test set to showcase what can be seen from both the cross-validation table and the matrices: the catchphrases sets, both train and test, don't yield results as good as the main test set. Besides the unavoidable difference in sizes of the sets, we argue that this is because the impact of catchphrases, at least when it comes to *Friends*, is not in their frequency, but rather in their memorability. They will seem to be linked specifically to a certain character because we as humans take into consideration so many other elements when hearing the catchphrase, from intonation and the situation the character is in, to our own opinion of the character. Joey, for example, said "How you doin'?" only several

[3] https://www.sbert.net/

[4] https://takelab.fer.hr/podium/index.html

[5] https://scikit-learn.org/stable/index.html

times in the show – and the classifier shows that – and yet we remember it as his signature line.

3.1. Statistical Significance Testing

To test our hypothesis about the overstated impact of catchphrases, we used statistical significance testing and SciPy's[6] *stats* module.

The null hypothesis was that our two populations (the accuracies obtained from the test set and phrases train set, respectively) are equal. Before testing the hypothesis, we first tested to see whether the two samples are normally distributed, since it influences the methods used in the next step. For this purpose we used two tests: Kolmogorov-Smirnov Normality Test and the Shapiro-Wilk Test. For both tests, we obtained a p-value significantly less than 0.05 for both samples and concluded that they deviate from a normal distribution. Thus, we used non-parametric tests for the next step.

To test our null hypothesis, we used the Mann-Whitney U Test. It yielded a p-value smaller than 0.05, which meant we reject our hypothesis that the two samples are equal; the model has a higher accuracy on the main test set than on the catchphrases train set (and, consequently, the catchphrases test set), as shown on 4 and in 3. We can conclude that catchphrases by themselves aren't as impactful to a trained classifier as one viewing the show might expect.

4. Conclusion

To summarize, we implemented a support vector machine model to tackle speaker identification and catchphrase resolution task. The classifier yielded good results for speaker identification, and somewhat worse results for the other dataset of catchphrases constructed with a paraphrase miner. We argued that this is because catchphrases don't have as much significance as a show viewer might think, and confirm our hypothesis with statistical significance testing.

Acknowledgements

We would like to thank Ms. Bhattacharyya for providing the *Friends* dataset we used for this task.

References

William M Campbell, Joseph P Campbell, Douglas A Reynolds, Elliot Singer, and Pedro A Torres-Carrasquillo. 2006. Support vector machines for speaker and language recognition. *Computer Speech & Language*, 20(2-3):210–229.

Timothy J Hazen, Douglas A Jones, Alex Park, Linda C Kukolich, and Douglas A Reynolds. 2003. Integration of speaker recognition into conversational spoken dialogue systems. In *Eighth European Conference on Speech Communication and Technology*.

Kaixin Ma, Catherine Xiao, and Jinho D Choi. 2017. Text-based speaker identification on multiparty dialogues using multi-document convolutional neural networks. In *Proceedings of ACL 2017, Student Research Workshop*, pages 49–55.

François Mairesse, Marilyn A Walker, Matthias R Mehl, and Roger K Moore. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of artificial intelligence research*, 30:457–500.

[6] https://docs.scipy.org/doc/scipy/reference/index.html

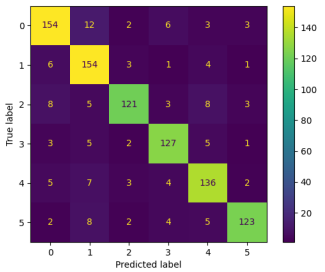


Figure 1: Confusion matrix for the main test set.

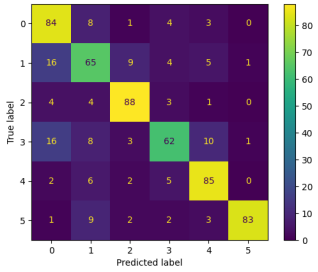


Figure 2: Confusion matrix for the catchphrases train set.

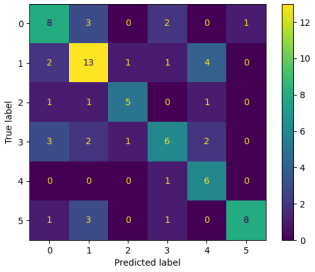


Figure 3: Confusion matrix for the catchphrases test set.

Table 2: Cross-validation results. A – accuracy, P – precision, R – recall. For gamma values: a – auto, s – scale.

hyperparameters			main test set				catchphrases train set				catchphrases test set			
kernel	C	γ	A	P	R	F1	A	P	R	F1	A	P	R	F1
RBF	1	s	0.795	0.799	0.794	0.796	0.712	0.723	0.712	0.711	0.494	0.494	0.502	0.480
RBF	1	a	0.191	0.0319	0.167	0.0535			—				—	
RBF	5	s	0.865	0.869	0.864	0.866	0.778	0.786	0.778	0.778	0.597	0.622	0.619	0.605
RBF	10	s	0.864	0.869	0.863	0.865	0.78	0.787	0.779	0.779	0.597	0.622	0.619	0.605
RBF	10	a	0.191	0.0318	0.167	0.054			—				—	
RBF	12	s	0.864	0.869	0.863	0.865	0.78	0.787	0.78	0.78	0.597	0.622	0.619	0.605
linear	1	—	0.500	0.502	0.497	0.498	0.442	0.463	0.442	0.44	0.377	0.37	0.367	0.349
linear	5	—	0.623	0.633	0.619	0.623	0.48	0.501	0.48	0.478	0.377	0.383	0.375	0.362
linear	10	—	0.677	0.687	0.674	0.678	0.49	0.503	0.49	0.487	0.429	0.448	0.443	0.417
poly	5	s	**0.866**	**0.871**	**0.865**	**0.867**	0.778	0.786	0.778	0.778	**0.597**	**0.622**	**0.619**	**0.605**
random			0.181	0.181	0.180	0.180	0.17	0.17	0.17	0.17	0.104	0.089	0.114	0.098
loudmouth			0.191	0.0318	0.167	0.0535	0.167	0.028	0.167	0.048	0.273	0.045	0.167	0.071

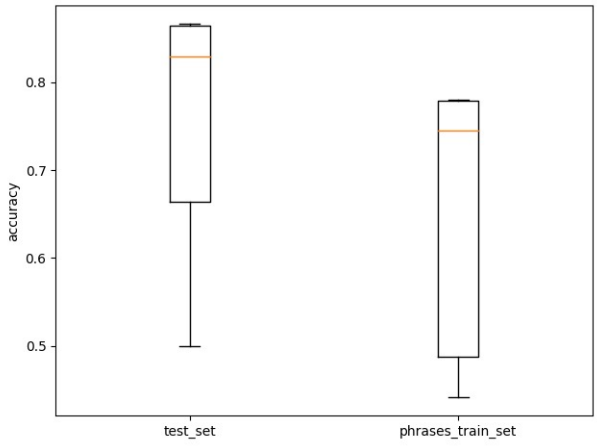


Figure 4: Box plot of the two sets. test_set here refers to the main test set.

Table 3: Results from statistical significance tests. K-S – Kolmogorov-Smirnov Normality Test, S-W – Shapiro-Wilk Test, M-W U – Mann-Whitney U Test.

	K-S	S-W	M-W U
Statistic	0.6914	0.6707	0.8097
p	0.0002	0.0004	0.0363

Dots, Do We Need Them? Significance of Punctuation in Irony Detection

Jelena Bratulić, Petar Kovač, Ivan Stresec

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{jelena.bratulic, petar.kovac, ivan.stresec}@fer.hr

Abstract

Sentiment analysis is the task of processing data with the goal of gauging general public opinion by classifying a text as either positive, negative or neutral. One particularly difficult aspect of sentiment analysis is irony detection since it can, and often does, change text sentiment. Text-based irony detection is a difficult problem owing to the additional textual cues that are often used and which can be ambiguous. In this paper, we discuss the effect of punctuation and punctuation-based features on the ability of a model to detect irony. The data we used is taken from the SemEval 2018 competition, task 3. We show that including punctuation substantially improves model performance on several neural network models commonly used in NLP (CNNs, RNNs and LSTMs).

1. Introduction

Irony is a form of figurative language that is often used to express the opposite of the literal meaning of a sentence. In speech, the irony is often delivered with a distinct tone of voice or a set of gestures that are not easily conveyed over text. This makes it difficult for machine learning models to discern ironic from non-ironic texts. Even for humans, this is not an easy task, owing to the complexity of language use and the need for context. Consider, for example, the following sentence:

I just love getting up early in the morning.

From the sentence alone it seems impossible to determine if this is a sarcastic sentence, without knowing the proper context. For example, if we knew the speaker normally never wakes up before noon, we would probably label this sentence as sarcastic. Sadly, this type of information is rarely available to us or is, at the very least, hard to get. Without contextual information, it is very hard to label such ambiguous sentences. Luckily, we could use other features which do not include context but could indicate sarcasm.

People naturally try to encode the way something was said into text, e.g., an angry person is more likely to write using capital letters. In this work, we argue that a sarcastic person is in a similar fashion prone to using specific punctuation to display sarcasm. The Merriam-Webster dictionary[1] defines punctuation as "*the act or practice of inserting standardized marks or signs in written matter to clarify the meaning and separate structural units*". Consider the previous sentence, but with additional punctuation:

I just "love" getting up early in the morning...

It seems to be more likely that this text is recognized as sarcastic, evident by the use of scare quotes and ellipses. In this paper, we seek to prove that omitting punctuation can have a significant impact on decreasing model performance when detecting sarcasm.

[1] https://www.merriam-webster.com/

The paper is organized as follows. Section 2 contains a brief overview of the current approaches to irony detection. In section 3 we examine the SemEval dataset and data preprocessing. Section 4 gives a summary of the models used, along with our methodology. In section 5 we present the results of our work and the hyperparameters with which they were achieved. We discuss the meaning of the results in section 6 and in section 7 we summarize the paper and propose avenues for future work.

2. Related Work

A lot of work has already been done regarding irony detection. Ghosh and Veale (2016) achieve notable results by using a model combining a Convolutional Neural Network (CNN) introduced by Lecun et al. (1998), a Long Short-Term Memory (LSTM) network introduced by Hochreiter and Schmidhuber (1997) and a Feed Forward Neural Network (FFNN). The authors collected a dataset of tweets among which many contained the #sarcasm or similar hashtag. These tweets were annotated as sarcastic, while tweets without sarcastic hashtags were annotated as non-sarcastic. Our models are very similar to the models presented in this paper and, to achieve comparable results, we also decided to use the dataset containing these hashtags.

Models which use handmade features can achieve impressive results, like the one that can be found in the work of Bouazizi and Ohtsuki (2015). The features used are sentiment-related, punctuation-related, pattern-related and finally lexical and syntactic features. It is interesting that the authors use hashtag information, but only to classify it as a positive or negative hashtag and to see if the hashtag sentiment differs from the sentence sentiment. Inspired by their use of features we tried to incorporate punctuation-based features in our models.

There is also reason to believe that research in the field of irony detection is incomplete without context (Wallace, 2015). It is argued that irony cannot be detected without proper context and that models using only shallow features cannot achieve state-of-the-art results. However, the approaches which are suggested completely differ from what

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 19–23, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

is presented in this paper and are only mentioned as an interesting alternative to models using shallow features.

3. Dataset

The dataset used in this paper is the SemEval 2018 task 3 competition dataset introduced by Van Hee et al. (2018). The dataset was extracted in the form of tweets using Twitter's[2] API. The competition contains two tasks: task A, in which the natural language processing (NLP) system should determine whether the tweet is ironic or not, and task B, where it is additionally necessary to determine which type of irony is used. The ironic tweets were collected using irony-related hashtags (i.e. #irony, #sarcasm, #not). As mentioned previously, we decided to use the dataset with the hashtags included.

In this paper, the task A corpus is the only one used. The training corpus consists of 3,834 tweets while the golden test corpus contains 784 tweets and both corpora have a balanced label distribution. The distribution of the positive and negative classes can be found in Table 1. The original dataset does not have a validation set so, to conduct model selection, we randomly sampled 20% of the training set for this purpose.

Table 1: Distribution of labels in the training and test corpora of the SemEval 2018 task 3 competition dataset A.

Set	Total	Positive	Negative
Train	3,834	1,911	1,923
Test	784	311	473

3.1. Preprocessing

The preprocessing pipeline consists of tokenization and feature extraction, for which we have used the Podium NLP library[3]. Other data handling was done using the NumPy[4], pandas[5] and scikit-learn[6] libraries.

For tokenization, we used the tweet tokenizer provided by the nltk library[7] along with the Podium preprocessing hooks for text cleanup (e.g. for punctuation removal). A regex replacement preprocessing hook was also used to break up long sequences of dots into separate ellipses. The regex replaces 2 to 5 dots with a single ellipsis and longer sequences of dots into separate ellipses. E.g., the sequence: "..... ...." will be replaced by " ... ... ".

The Podium post-tokenization hooks were used to extract punctuation-based features from the tweets. These include the number of dots, question marks, exclamation marks, quotation marks, ellipses and the total number of interpunction characters used. These features are not always extracted and the use of features depends on the configuration of the model. Likewise, punctuation is also not always used and can be removed in preprocessing.

Podium was additionally used for vocabulary creation and numericalization. The vocabulary was restricted to 10,000 words and made using the training part of the dataset (excluding validation). This vocabulary was then fed to a GloVe[8] (Pennington et al., 2014) loader to create an embedding matrix. Before vectorization, tweet tokens were padded to a fixed length that is equal to the number of tokens in the longest token sequence (37 without punctuation, 40 with punctuation).

4. Our Approach

In this paper, our goal was not obtaining state-of-the-art results in irony detection, but rather determining if punctuation has a significant impact on the effectiveness of irony detection classification models, or not. We wanted to test the impact of punctuation on several neural network models that are commonly used in NLP. That being said, we still perform hyperparameter selection using a validation hold-out set since we deem results on ineffective and unoptimized models less relevant. Hyperparameter search was conducted manually and selection was based on the validation set scores. The hyperparameters of the model with the highest F_1 score on the validation set without punctuation and features were used. We deemed this approach to be the most unbiased, as optimizing the models using punctuation could create an unfair advantage.

All models use 300-dimensional GloVe embeddings, which provide good preliminary vector representations of words. When used, punctuation-based features were simply concatenated to the models' inputs. Since the features carry no local information, in recurrent and convolutional models the features were concatenated after those layers, as an input to the decoder (the FFNN). All neural network models were implemented using PyTorch[9] and for the baseline, we used scikit-learn's Support Vector Machine (SVM). The training of PyTorch models was done using PyTorch's implementation of the Adam optimizer (Kingma and Ba, 2015). For regularization, we have used weight decay (L_2 regularization) as well as early stopping using the validation set. Values of these and other optimized hyperparameters not relating to architecture are shown in Table 2.

In the following subsections, we further describe the models we used and tested. The source code can be found in our GitLab repository[10].

4.1. Baseline SVM

For a baseline, we used a simple SVM with a radial basis function kernel which took the average of GloVe representations of a tweet as input. The regularization parameter used was $C = 1$. SVMs are often used as baselines due to their simplicity and ability to handle high-dimensional data relatively well.

[2] https://twitter.com/
[3] https://takelab.fer.hr/podium/
[4] https://numpy.org/
[5] https://pandas.pydata.org/
[6] https://scikit-learn.org/stable/index.html
[7] https://www.nltk.org/
[8] https://nlp.stanford.edu/projects/glove/
[9] https://pytorch.org/
[10] https://gitlab.com/istresec/irony-detection

Table 2: Optimized model hyperparameters. ES epochs stands for early stopping criteria epochs. The weight decays are written for FFNN, GRU-FFNN, CNN-FFNN and BiLSTM-FFNN (same value for both), and CNN-LSTM-FFNN, respectively.

Hyperparameters	Value
Batch size	32
Learning rate	10^{-4}
GloVe dimension	300
ES epochs	50
Dropout	0.2
Weight decays	10^{-3}, $2 \cdot 10^{-4}$, $5 \cdot 10^{-4}$, $5 \cdot 10^{-5}$

4.2. FFNN

The first NN model we used is a simple FFNN model with 2 fully connected (FC) layers and the rectified linear unit (ReLU) function as a nonlinearity. The first layer has 300 hidden units and the second, output layer has 2 outputs – the logits.

4.3. CNN-FFNN

The CNN-FFNN is an upgraded FFNN model which uses 2 additional convolution layers before the FFNN, with the first FC layer having 150 hidden units this time around. Convolution layers are introduced with the idea of modelling local relations between words. The first layer is 1D and its kernel is of size 3×300 (300 being the GloVe embedding dimensionality) and the input is not additionally padded. The second kernel is 1D of size 3 and uses padding. These kernel sizes were shown to be effective by Ghosh and Veale (2016) and we did not optimize them further.

4.4. GRU-FFNN

GRU-FFNN is a simple recurrent model with a Gated Recurrent Unit (GRU) and a decoder FFNN with $tanh$ as the nonlinearity and 300 hidden units in the first FC layer. GRUs were introduced by Cho et al. (2014) and are similar to LSTMs but have fewer parameters. They are often successfully used in NLP. Our model uses 2 GRU layers of 300 hidden units. The input is used without padding since a Recurrent Neural Network (RNN) can handle variable-size input.

4.5. BiLSTM-FFNN

Much like GRU-FFNN, BiLSTM-FFNN is a recurrent model with bidirectional LSTM units and a FFNN decoder. LSTMs are canon in their NLP usage and are often a part of state-of-the-art architectures for many tasks. The input is used without padding just like in GRU-FFNN and only the last hidden state of the last layer is used as input for the FFNN – this is common practice. Here the FFNN uses ReLU again and its first FC layer has 50 hidden units and we use 2 BiLSTM layers, also with 50 hidden units.

4.6. CNN-LSTM-FFNN

This model was implemented according to the one introduced by Ghosh and Veale (2016) and is a sort of amalgam between the CNN-FFNN and LSTM-FFNN models. We decided to use this arguably more complicated architecture to have performance information on a relevant irony detection model. The kernel sizes and paddings for the convolution layers are the same as in the CNN-FFNN model. The input to the FFNN is the whole hidden state of the last layer of the LSTM and the nonlinearity between FFNN layers is the logistic sigmoid, like in the original model. A graphic representation of the model can be seen in Figure 1.

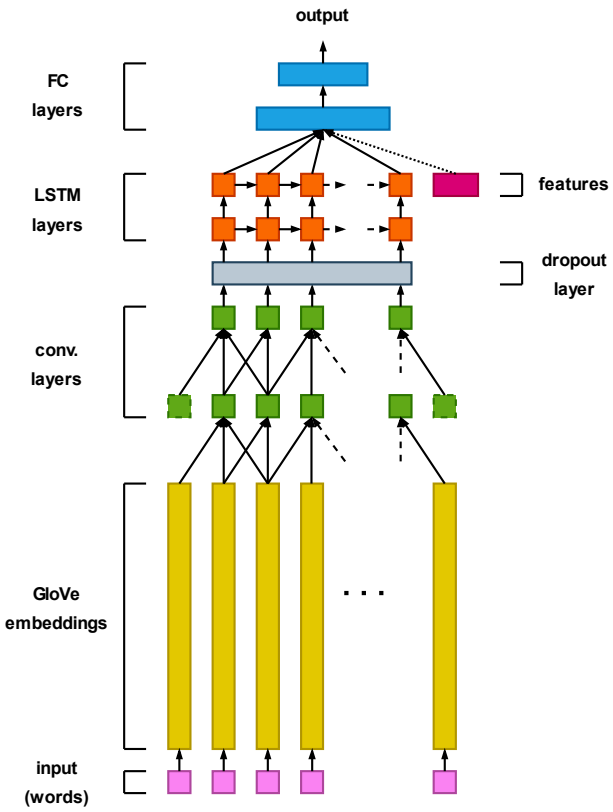


Figure 1: A graphic representation of the CNN-LSTM-FFNN architecture, with added features.

5. Results

In this section, we evaluate previously introduced models. Each model was trained with three different modes. In the first mode (*No Punctuation*), the inputs were preprocessed by removing all punctuation from the input tweets. In the second mode (*Punctuation*) the tweets were not modified – the punctuation was kept and normalized, as explained in the Preprocessing section. Finally, the third mode (*Punctuation and Features*) kept the original tweets with punctuation and used additional punctuation-based features.

All models were evaluated using accuracy, precision, recall and the F_1 metric. Even though the label distribution of the dataset is even, we used the F_1 metric as it has been often used in other papers which deal with the task of irony detection. Additionally, we also calculated Matthew's correlation coefficient (also known as Pearson's phi coeffi-

Table 3: Metrics evaluated on the test set for different models and training modes. Acc stands for accuracy and MCC stands for Matthew's correlation coefficient. † denotes punctuation significance and ‡ denotes feature significance.

Model \ Mode	No Punctuation			Punctuation			Punctuation and Features		
–	Acc	F_1	MCC	Acc	F_1	MCC	Acc	F_1	MCC
SVM†	0.7781	0.6926	0.5279	0.8737	0.8533	0.7508	0.8890	0.8651	0.7725
FFNN†	0.8037	0.8006	0.6566	0.8538	0.8567	0.7598	0.8638	0.8692	0.7841
CNN-FFNN†‡	0.8163	0.7988	0.6583	0.8925	0.8920	0.8180	0.8688	0.8603	0.7654
GRU-FFNN†	0.8688	0.8629	0.7682	0.8950	0.8976	0.8279	0.8800	0.8730	0.7879
BiLSTM-FFNN†‡	0.8538	0.8434	0.7356	0.8862	0.8851	0.8062	0.8475	0.8389	0.7262
CNN-LSTM-FFNN†	0.8562	0.8555	0.7550	0.8938	0.8906	0.8173	0.8875	0.8872	0.8096

cient), which is argued to be more informative and truthful than F-scores, as well as having some other good properties (Chicco and Jurman, 2020; Baldi et al., 2000). Table 3 shows several metrics on the test set for different models and modes. We can see that the *Punctuation* mode outperformed the *No Punctuation* mode for all of the models.

Furthermore, we decided to use McNemar's test (McNemar, 1947) as we wanted to test a lot of the models which took relatively long to train on the hardware available to us. This also allowed us to use the same test set provided in the SemEval dataset for each model, which was convenient. McNemar's test, in the case of comparing two binary classifications, tells us whether the two models disagree in the same way, or not. It is worth noting, however, that McNemar's test does not provide us with the information on whether one model is more or less accurate than the other.

McNemar's test showed that there is significant difference in class disagreement with the significance level of 5% when testing the first and second mode for all the models which backs up our hypothesis. Moreover, when we look at the impact of punctuation-based features, we can see that they even have a negative impact on the model, compared to only punctuation being used. McNemar's test on the significance of features showed that there is no significant disagreement between classes for all the models except BiLSTM-FFNN and CNN-FFNN models.

6. Discussion

Text-based irony detection is a difficult problem owing to the additional textual cues that are often used. We, as human annotators recognize the significance of these cues when deciding if a sentence is ironic. However, we wanted to find out if machine learning models could successfully use this information, specifically punctuation, as well.

Our results demonstrate that the impact of using punctuation is there – models that do not remove punctuation consistently outperform those which do. Using McNemar's test we even showed statistical significance of different classification when using punctuation on all of the used models. Therefore, in our opinion, removing punctuation could unnecessarily remove valuable semantic information from a text and this should not be done in irony detection and possibly in some other sentiment-based NLP tasks as well. Models could also overfit the significance of punctuation given unbalanced datasets and lack of regularization, but we had no such problems. Using weight decay as well as early stopping produced good results without any signs of such errors.

We also tried using simple punctuation-based features to see if our models could exploit basic numerical information about the used punctuation, but this failed with all but one model (FFNN). We suspect that this could be due to the features being overly specific and the dataset being relatively small, allowing only for overfitting without any useful information extraction.

In future work, we hope to further reinforce our findings using different datasets such as the dataset used by Riloff et al. (2013), as well as datasets for other languages such as the one used by Ptáček et al. (2014). We are also interested in the influence of punctuation in non-binary classification problems relating to irony like task 3-B of the SemEval 2018 competition, as well as testing the effect of punctuation on newer models that use Bidirectional Encoder Representations (BERT), which often produce state-of-the-art results (Devlin et al., 2019). Last but not least, we would like to examine the performance of models with the irony-related hashtags removed, as they might introduce unwanted bias.

7. Conclusion

In this paper, we examined the significance of punctuation in the task of irony detection. We have tested several different neural network models widely used in NLP including CNNs, RNNs and LSTMs, as well as a baseline SVM model. GloVe embeddings were used as inputs and fine-tuned during training. The models were trained on the Twitter-based SemEval 2018 task 3A dataset with three different modes: *No Punctuation*, *Punctuation*, and *Punctuation and Features*.

Our results show that the models which use punctuation outperform the models which do not, confirming our intuitions about the importance of punctuation in ironic statements. Punctuation-based features did not improve models which already used punctuation, probably due to their overly specific form and a relatively small dataset.

Acknowledgements

We would like to thank Josip Jukić for giving useful advice along the way and answering our questions, of which there were many.

References

Pierre Baldi, Søren Brunak, Yves Chauvin, Claus Andersen, and Henrik Nielsen. 2000. Assessing the accuracy of prediction algorithms for classification: An overview. *Bioinformatics (Oxford, England)*, 16:412–24, 06.

Mondher Bouazizi and Tomoaki Ohtsuki. 2015. Sarcasm detection in twitter: all your products are incredibly amazing!!! - are they really? In *2015 IEEE Global Communications Conference, GLOBECOM 2015*, 2015 IEEE Global Communications Conference, GLOBECOM 2015. Institute of Electrical and Electronics Engineers Inc. 58th IEEE Global Communications Conference, GLOBECOM 2015 ; Conference date: 06-12-2015 Through 10-12-2015.

Davide Chicco and Giuseppe Jurman. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1):6, January.

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. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 161–169, San Diego, California, June. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780, November.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, June.

Jeffrey Pennington, Richard Socher, and Christopher 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, Doha, Qatar, October. Association for Computational Linguistics.

Tomáš Ptáček, Ivan Habernal, and Jun Hong. 2014. Sarcasm detection on Czech and English Twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 213–223, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 704–714, Seattle, Washington, USA, October. Association for Computational Linguistics.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. SemEval-2018 task 3: Irony detection in English tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50, New Orleans, Louisiana, June. Association for Computational Linguistics.

Byron C. Wallace. 2015. Computational irony: A survey and new perspectives. *Artificial Intelligence Review*, 43(4):467–483, April.

An Active Learning System for Quora Duplicate Question Detection

Rino Čala, Marin Petričević, Ante Pušić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{rino.cala,marin.petricevic,ante.pusic}@fer.hr

Abstract

Abundance of data is one of the driving factors behind the deep learning revolution, but for specific industry use cases, datasets as large as the commonly available academic datasets can be very expensive. We approach this problem using active learning, a method guiding data annotation as to label the most informative data points. In this paper, we describe our active learning based system, where we simulate a lack of labeled data on the Quora Duplicate Question dataset and utilize a maximum uncertainty heuristic for labeling selection. This system could be useful for a competitor looking to kick-start their machine learning based moderation system.

1. Introduction

The process behind solving a novel problem with a machine learning system is quite complex, starting with a potentially expensive process of data annotation. Approaching a novel problem presents engineers with many unknowns regarding the size of the dataset needed and the ultimate limit of current machine learning algorithms on that problem. To deal with the issue, a good practice is to carry out experiments during the process of dataset creation. Implemented models can help determine the need for additional annotations and provide a sanity check of the whole process.

When gathering annotations, it is useful to have an indication of which examples should be annotated next. This can reduce the cost of the whole process, result in better resource distribution and better performing models that are being developed. One of the ways to do this is by carrying out active learning during the annotation phase. The model, which is trained on a small initial set of annotated examples, determines which examples from the pool of unannotated examples will be annotated next. Priority will be given to those examples that the model deems most informative. Using active learning, in just a few annotation rounds, a well-performing model and a high-quality informative dataset can be obtained, all while labeling significantly fewer data points in total.

Using the Quora Duplicate Question dataset we will demonstrate the usability of active learning by simulating an unannotated dataset which we progressively annotate in a simulated active learning process. Our experiments are conducted on two models representing the two major approaches a team with limited compute resources can use, a trained from scratch LSTM model, and a fine-tuned Transformer model. We used a two-layer BiLSTM model and a BERT based model, which were trained on a small subset of the dataset and were tasked with predicting which examples should be added to the subset for the next iteration of active learning. We demonstrate the performance of the models in every iteration of active learning and provide a fair comparison with the random sampling of new examples to argue the effectiveness of using active learning.

2. Related Work

The Quora Duplicate Question dataset was initially presented as a Kaggle competition[1] with cash prizes. As such, there is a large body of highly competitive models. All the top models in the competition were large ensembles with hundreds of models and thousands of features[2][3]. While we have no ambition of replicating those complex models, they are useful as a measure of the upper bound of the dataset. Unfortunately, the top submissions to the competition only mention their final log loss score on the private leaderboard test set. We do not have access to those labels, so we can't directly compare our models, but we can get a feeling for the upper bound from validation accuracies of not quite winning, but still well ranked teams. We look to the 14th ranked submission[4] which reports a validation accuracy of 0.8652 with a single relatively big LSTM based model.

Generally, our problem is considered a sentence pair classification problem, which is approached similarly to natural language inference. Modern examples of approaches include using siamese or triplet networks to generate sentence embeddings which can then be compared for similarity using cosine distance in order to detect semantically similar sentences (Reimers and Gurevych, 2019). It can also be approached as a binary classification problem, either trained directly, or as a classification head on top of previously mentioned embeddings, like Homma et al. (2016).

On the active learning front, we used the Human in the Loop Machine Learning book by Munro (2020) as a guiding hand through the field of active learning. The specific method of uncertainty sampling from a pool of unlabeled data to guide annotation, that we ended up using in our system was first presented by Lewis and Gale (1994). This method has stood the test of time as a simple and effective way to do active learning.

[1] https://www.kaggle.com/c/quora-question-pairs

[2] https://www.kaggle.com/c/quora-question-pairs/discussion/34355

[3] https://www.kaggle.com/c/quora-question-pairs/discussion/34310

[4] https://github.com/Wrosinski/Kaggle-Quora

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 24–27, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

Table 1: Example Datapoints

question1	question2	is_duplicate
Who created the periodic table?	What was the reason the periodic table was created?	F
How do you self publish a book?	How can I publish my own book?	T
What separates a good programmer from a bad programmer?	What separates a good programmer from a great one?	T

3. Dataset

The Quora Duplicate Question dataset consists of 404290 unique question pairs consisting of 537933 unique questions. This means that certain questions appear in multiple question pairs. Of all the question pairs, 63.08% are labeled as not-duplicates, while 36.92% are labeled as duplicates. This means we are dealing with a slightly unbalanced dataset. The dataset, besides id fields for questions and question pairs, contains two text fields containing the two questions, as well as a binary label field. Examples from the dataset are presented in Table 1.

We note that the dataset is somewhat noisy, both in the questions and in the labels. Being sourced from a diverse set of real social media users, there are naturally some typos and slang, but we generally find this to be not too bad. More problematic is the noise in the labels. Cursory examination of the dataset easily results in question pairs where the authors do not agree with the assigned labels. One such example is given in the third row of Table 1, which is assigned as duplicate, but we feel should have been labeled as not duplicate. We make no attempt at estimating this labeling error, but we suspect it is significant enough to limit final model accuracy.

Since we do not have access to the labels for the provided test set, we construct our own as part of the standard $70/15/15$ train/validation/test split. We consider the possibility of the same questions appearing in train and test to not be an instance of information leakage, since the question pairs themselves are unique and we do not approach the problem from a clustering, but a classification point of view.

4. Models

Following the top ranked solutions on the Kaggle competition[5][6], we decided to use a LSTM architecture (Hochreiter and Schmidhuber, 1997), specifically a two-layer BiLSTM model like Sharma et al. (2019).

When feeding two questions to the model, we used GloVe word embeddings (Pennington et al., 2014) of dimension 300 as the vector inputs to our model. We decided upon a BiLSTM hidden size of 150 and feed the last hidden size of the last layer of the BiLSTM model to the linear layer of size 150 and a softmax layer predicting if the two questions are paraphrases of each other.

During experiments, we found that the model best performs on the validation set after training it for 5 epochs on the training set with the batch size of 32. We used a learning rate value of 0.002 with a linear decay over the 5 epochs. For optimization, we used an Adam (Kingma and Ba, 2017) optimizer which performed the best in our case.

Another model that we used for our experiments is BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019). When feeding the two questions to the model we concatenated them with a [SEP] token and added a [CLS] token at the beginning. To get the prediction for the concatenated questions we used a Huggingface (Wolf et al., 2020) implementation of the BERT model which has a linear layer over the pooled output suited for classification tasks.

We managed to get the best results with the model on the validation set after fine-tuning it with a batch size of 32 for 3 epochs on the train set using an Adam optimizer.

5. Active Learning

Our Active Learning setup is a fairly simple one. All presented experiments start with 5660 randomly chosen question pairs, which is 2% of the total train split. These form the train set for the initial model. That model is then used to estimate the most informative question pairs from the pool of remaining question pairs in the train split, which for the first iteration is 98%, or 277342 question pairs. The algorithm chosen to estimate most informative question pairs is least confidence sampling, which is a form of uncertainty sampling. This means we look at the difference between the most probable label according to our model and 100% probability. Since our problem is binary classification, we effectively search for question pairs for which our model predicts 50-50 chance for a specific pair of questions being duplicates, or rather we search for pairs where the predictions are as close as possible to 50-50. We experiment with two rates of data annotation per active learning round, adding 4040 or 8080 per round. These numbers correspond to 1% and 2% of the total train set.

Least confidence sampling is a tried and tested method of active learning. Its simplicity is a great advantage. The fact that only final model predictions are necessary makes it applicable to basically every machine learning model. The simplicity also means it's quick and easy to implement, which when looked at from a budget allocation perspective means money which could be used to pay annotators is not being used by engineers devising advanced and complicated active learning schemes.

The main disadvantage of least confidence sampling is

[5] https://www.kaggle.com/c/quora-question-pairs/discussion/34288

[6] https://www.kaggle.com/c/quora-question-pairs/discussion/34697

Table 2: Accuracy difference between active learning and random, full dataset benchmark

Data	BiLSTM 2%			BERT 2%		
	Test acc	Δrandom	Δbenchmark	Test acc	Δrandom	Δbenchmark
$\sim 10\%$	0.720	0.005	0.082	0.850	0.01	0.044
$\sim 15\%$	0.728	0.032	0.074	0.861	0.005	0.033
$\sim 20\%$	0.752	0.055	0.050	0.877	0.018	0.017
$\sim 25\%$	0.747	0.013	0.055	0.881	0.017	0.013
$\sim 30\%$	0.761	0.030	0.041	0.889	0.024	0.05

Table 3: Results on test and validation trained on entire train split

Model	Val	Test
Majority	63.4	63.2
BiLSTM	80.0	80.2
BERT	89.8	89.4

that in the exploration vs. exploitation dichotomy, least confidence sampling is entirely biased on the side of exploitation. This entails a risk that a model may choose suboptimal data points to be labeled, which can lead to stagnation or even reductions in performance, despite a larger training dataset, when evaluated on a hold-out test set. This phenomenon is similar to classic over-fitting in machine learning, but is not caused by too large of a model capacity, but by a bias in data collection, which causes our training dataset to not be a representative sample of the input population, but a largely biased one.

6. Results

The results of our two models on the validation and test set are shown in Table 3. Both of the models were trained on the train set with BERT getting far better performance over the BiLSTM model. This is expected considering that BERT is a much larger model pre-trained on a large language modeling task, whereas our BiLSTM model has a much smaller size and is trained on the Quora Duplicate Question dataset only. The accuracies of these models will represent an upper bound benchmark for our active learning system considering that they are obtained after training the models on the whole train split. We will use the accuracy of the Majority classifier as a baseline and sanity-check for the active learning system.

For the BiLSTM model, as can be seen in Figure 1., the results are a minor success. Of the two active learning regiments, the one with more new data per active learning step (2%) is significantly more stable and generally performs better. Table 2 shows that the results are no worse when using active learning at any point, and at most points, active learning increases accuracy by a few percent.

The model based on BERT, shown in Figure 2., performed significantly better in the active learning regime, with both very stable increases in performance as it is given

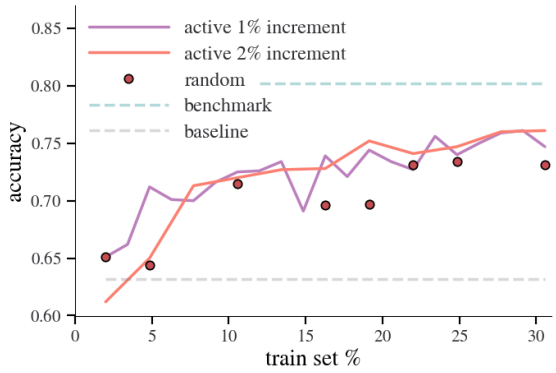


Figure 1: Active learning results using the BiLSTM model

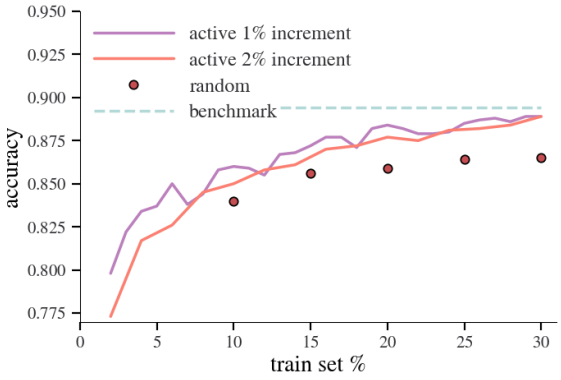


Figure 2: Active learning results using the BERT model

more data, while showing consistent difference from random sampling. As a function of data, the BERT based model not only shows a steeper trend of improvement compared to random sampling, but when using 30% of the train split data it only has a 0.005 lower accuracy than the exact same model trained on the entire train split.

Table 2 showcases that the BERT based model approaches the full dataset accuracy much more quickly compared to the BiLSTM model. We suspect that this is because the BiLSTM model is trained from scratch and therefore, it has no inherent knowledge of language beyond the word embeddings, and the few thousand question pairs that

it is given at the start of the active learning process seem to be not enough for the model to be able to grasp the task enough to be able to direct active learning much better than random search. This may be the cause of the instability seen in Figure 1. We suspect a larger initial dataset would be useful for models trained from scratch, so as to start the active learning process with a larger exploration base. The model is also likely over-parameterized for the very small initial datasets, so greater stability may be possible with a smaller model, but this may negatively impact final accuracy.

The BERT model on the other hand is already trained for language modeling, this means it already has a decent idea of what language is, and only has to slightly adapt to match this new domain. This base knowledge also seems to provide the model with a great ability to direct the active learning process. These results indicate that active learning when combined with pre-trained models, can be a very data efficient way of building classifiers, possibly requiring only 30% of the data that would be required if the data was just randomly annotated.

7. Conclusion

In this paper we have presented our system for active learning on the Quora Duplicate Questions dataset. We confirm that very simple active learning setups, using maximum uncertainty sampling for annotation candidate selection, can be an effective way to reduce annotation costs and quickly produce high quality models. We show that active learning is not equally effective in a pre-trained and trained-from-scratch model, showing much clearer gains when using a pre-trained model. Reassuringly, in the trained-from-scratch BiLSTM model, the active learned model never performed worse than the one trained on equivalent random data.

Active learning for classification when using a pre-trained BERT model showed great data efficiency, reaching the full model accuracy with only 30% of the data, thus presenting itself as an obvious way to affordably increase model performance when solving a novel task of this type.

Possible future work includes additional experimentation to estimate variability in active learning and random sampling performance, which was excluded due to time and/or compute constraints. Additionally more complex active learning systems would be interesting, particularly diversity sampling methods, which are more biased to exploration than exploitation, compared to uncertainty sampling. This sampling method would be particularly interesting in combination with siamese and triplet networks, which naturally produce embeddings. Last but not least, data cleaning could prove very useful, either by, for example, applying spelling correction, or by fixing mislabeled data.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80, 12.

Yushi Homma, Stuart Sy, and Christopher Yeh. 2016. Detecting duplicate questions with deep learning. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS*.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer.

Robert Munro. 2020. *HUMAN-IN-THE-LOOP MACHINE LEARNING*. OREILLY MEDIA.

Jeffrey Pennington, Richard Socher, and Christopher 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, Doha, Qatar, October. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Lakshay Sharma, Laura Graesser, Nikita Nangia, and Utku Evci. 2019. Natural language understanding with the quora question pairs dataset.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Text Augmentation: Does it Make Sense?

Marko Čuljak, Darijo Brčina, Vedran Kolka

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{marko.culjak,darijo.brcina,vedran.kolka}@fer.hr

Abstract

Learning commonsense knowledge is a difficult task for machines. However, with the recent advent of transformer models, the ability of NLP systems to learn commonsense knowledge has significantly improved. Motivated by the lack of attention to text augmentation approaches in tackling this challenge, in our research, we experiment with several text augmentation techniques, including a task-specific GPT-2-based technique. We train multiple models in order to explore the effect of text augmentation on the performance of commonsense knowledge validation systems. Our best model yields results comparable to state-of-the-art, with an accuracy of 95.6% on the SemEval-2020 Task 4, Commonsense Validation, subtask A test set.

1. Introduction

Most adult humans are equipped with common sense i.e., according to Cambridge Dictionary, the ability to use good judgment in making decisions and to live in a reasonable and safe way. Even though commonsense knowledge is mostly basic, it is not a fundamental principle of the Universe but rather a collection of knowledge built by generations of humans through observations of phenomena in their surroundings. Since humans are not born with common sense, they start learning it from the earliest stages of their life, either explicitly or implicitly.

On the other hand, teaching computers commonsense knowledge is a difficult task. This is primarily due to two reasons: (1) unlike humans, machines are unable to learn implicitly or passively and (2) commonsense knowledge is broad and constantly evolving. Yet, recently, with the advent of transformer models (Vaswani et al., 2017), the ability of NLP systems to learn commonsense knowledge has improved significantly. The approaches based on fine-tuning transformer models previously pre-trained on large text corpora continuously achieve state-of-the-art results on natural language understanding (NLU) tasks. However, not much attention has been paid to text augmentation, especially in NLU tasks.

In this paper, we tackle the first subtask from SemEval-2020 Task 4, Commonsense Validation and Explanation (ComVE) (Wang et al., 2020) by employing several text augmentation techniques. We explore the effect of these techniques on the performance of common sense validation systems. We experiment with back translation and contextual embeddings-based data augmentation techniques. Furthermore, we propose a text augmentation technique that leverages a GPT-2 model (Radford et al., 2019) to generate artificial data. The subtask on which we evaluate the models' performance requires them to identify which of the given two sentences does not make sense. For example, for the following two sentences:
s_0 : *He poured milk on his cereal.*
s_1 : *He poured orange juice on his cereal.*
the desired output of the model is 1 because the sentence $s1$ is against common sense. The data provided as a part of the task is described in Section 3. The systems, along with text augmentation techniques we experimented with are described in detail in Section 4. and Section 5., respectively. Finally, in Section 6., we present the results of the experiments.

2. Related Work

In recent years, many approaches tackle common sense-making and reasoning leveraging transfer learning and transformer models. Wang et al. (2019) present promising results on common sense-making and not as promising results on common sense reasoning. The authors state that the results show that sense-making remains a technical challenge for such models, whereas inference is a key factor that is missing. Nevertheless, the results are meaningful because they show that artificial intelligence systems are still far from human-like performance in common sense reasoning.

The most successful approaches to the Validation subtask (Zhang et al., 2020; Zhao et al., 2020) leverage pre-trained transformer-based models and external commonsense knowledge stored in the ConceptNet (Liu and Singh, 2004). However, as stated by Wang et al. (2020), the success of the ConcepNet-based systems may be attributed to the data leakage which may have occurred because ConceptNet was used as an inspiration for ComVE data generation along with transformer-based models. Rather than leveraging the existing knowledge bases, we attempt to generate additional knowledge artificially by employing text augmentation techniques.

Two approaches at ComVE employed back translation to enhance their common sense validation model performance (Liu et al., 2020; Jon et al., 2020). Although they reported an improvement over their baselines, they do not back their results with statistical significance testing. As for GPT-2 artificial data generation, our approach is inspired by Kumar et al. (2020).

3. Dataset

ComVE subtask A dataset includes 10000 sentence pairs in the training set, 997 samples in the validation set, and 1000

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 28–32, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

Table 1: The average word counts of sensical and non-sensical sentences.

	Training	Validation	Test
Sensical	7.67	7.12	7.25
Non-sensical	7.69	7.16	7.36

samples in the test set. The distribution of labels is balanced. The average word count of sensical and non-sensical sentences in the training, validation, and test set is given in Table 1.

4. Common Sense Validation Model

Our system consists of a pre-trained transformer model as a sentence encoder which is followed by a dropout layer, a linear layer, and a softmax layer. The linear layer is used to map the sentence encoder's output to a single score value. The softmax layer is used as a score comparator. The architecture of the proposed system is visualized in Figure 1.

As encoders we used the following transformer models: DistilBERT$_{\text{BASE}}$ and DistilRoBERTa$_{\text{BASE}}$ (Sanh et al., 2020), ALBERT$_{\text{BASE}}$ (Lan et al., 2020), ELECTRA$_{\text{SMALL}}$ and ELECTRA$_{\text{BASE}}$ (Clark et al., 2020). Due to resource limitations we optimize the hyperparameters only on the model based on ELECTRA$_{\text{SMALL}}$ and reuse the obtained hyperparameters on other transformer-based models. For the same reason we do not consider the largest versions of the aforementioned transformers. Aside from transformers we also experimented with a Bidirectional LSTM (BiLSTM) (Schuster and Paliwal, 1997) as the encoder.

4.1. Implementation Details

The transformer models were downloaded through HuggingFace transformers (Wolf et al., 2020) module. Fine-tuning of our model was done using Pytorch Lightining[1] framework in Google Colab[2] environment.

We trained our models using the AdamW optimizer. After a grid search hyperparameter optimization with respect to the provided validation corpus, we concluded the optimal parameters for the AdamW optimizer are 4e-5 for the learning rate, and 1e-2 for the weight decay. We used a linear scheduler with warm-up for adjusting the learning rate during training and we set the number of warm-up steps to correspond to one epoch. Following the same hyperparameter optimization procedure, we concluded that the optimal batch size is 32. As for the dropout layer, we set drop probability for systems trained solely on original data and data augmented by employing back translation and contextual embeddings to 0.1, while for systems that leverage data generated by GPT-2 we set the drop probability to 0.8. We trained our models through 8 epochs, saving the best model with respect to validation loss.

Since the BiLSTM-based model performed poorly on the validation set, we were unable to find hyperparameter values that stand out. We settle for an architecture with 2 BiLSTM layers, a hidden size of 300, a learning rate of 5e-4, and weight decay of 1e-2. As for word representations, we use 300-dimensional GloVe embeddings (Pennington et al., 2014).

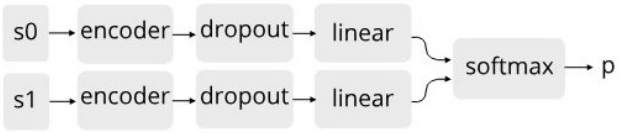


Figure 1: The system architecture used in all experiments. The *encoder* is either a transformer or a BiLSTM and p denotes the predicted probability that s_1 is against common sense given s_0 and s_1 as inputs.

5. Text Augmentation

Classic text augmentation techniques are based on deletion, addition, and replacement of certain words as well as changing the word order. Such techniques are not useful for NLU tasks because they may render sensical sentences non-sensical and vice versa. For example, if we change the word order in a sentence *A man eats a cake.* we may end up with a sentence *A cake eats a man.*, which is obviously against common sense. On the contrary, techniques based on alternating single letters in a text are also not particularly useful for NLU tasks since they do not modify the text in a manner significant enough to create new knowledge, although they may contribute to the robustness of a system to spelling errors, which is not essential for such tasks.

However, techniques such as back translation (BT) and replacement of certain words based on contextualized word embeddings (CWE) do not affect the sense of the sentences strongly and may be able to generate additional knowledge, which is why we find them worth exploring. Aside from only modifying the provided data, we attempt to generate artificial sentence pairs by leveraging a GPT-2 model fine-tuned on the provided training data.

We used NLPAug[3] library to perform back translation and CWE-based text augmentation and HuggingFace transformers to obtain the GPT-2 model. After obtaining the artificially generated datasets, we concatenated them to the original corpus, removing the duplicated pairs.

5.1. Back Translation

The back translation text augmentation technique leverages two machine translation models: (1) a model that translates text written in language A to language B and (2) a model that translates the obtained text back from language B to language A. The main motivation for back translation is the fact that texts may have multiple valid translations. In this work, we choose German as a language B.

5.2. Contextualized Word Embeddings

CWE text augmentation leverages contextualized word embeddings to find a replacement of a word based on its surroundings. The hyperparameters of CWE text augmentation are thoroughly described in NLPAug's documenta-

[1] https://github.com/PyTorchLightning/pytorch-lightning

[2] https://colab.research.google.com/

[3] https://github.com/makcedward/nlpaug

Table 2: Accuracies(%) of GPT-2_{LARGE} with harmonic mean (HM), arithmetic mean (AM), geometric mean (GM), product (Prod) aggregation methods on the validation set. The best two results are in bold.

HM	AM	GM	Prod
65.0	66.4	**74.9**	**78.8**

Table 3: Word count means of the first 5k artificially generated sentences in GPT-2-G and GPT-2-P datasets. The average sentence lengths for the GPT-2-G are similar to those of the original training set.

	GPT-2-P	GPT-2-G
Sensical	4.79	8.34
Non-sensical	5.00	7.82

tion[4]. In this work, we use $\text{BERT}_{\text{BASE}}$ for computing the CWE. We modified only one word in each sentence in a pair and set `top_k` to 30 and `temperature` to 0.2. The `top_k` and `temperature` parameters are selected by performing a grid search and manually evaluating the quality of 20 generated sentences.

5.3. GPT-2

Our approach in leveraging GPT-2 for text augmentation can be broken down into four phases: (1) fine-tuning, (2) generation[5], (3) preprocessing, and (4) selection.

Fine-tuning. Firstly, we fine-tune the GPT-2_{MEDIUM} model on the corpus of concatenated sentence pairs obtained from the ComVE training dataset. Each text in the corpus begins with an "S:" string to denote the start of a text, followed by a sensical sentence, "/" token, a non-sensical sentence, and the "<| endoftext |>" token. We fine-tune the model for 3 epochs, with a learning rate of 2e-5 scheduled by the cosine scheduler with warm-up with 300 warm-up steps, and with the batch size of 32.

Generation. In the generation phase we use the fine-tuned model to generate three times as many sentence pairs as the size of the training dataset (30k sentence pairs). For text generation, we use nucleus sampling (Holtzman et al., 2019). This sampling technique randomly selects the next token from the smallest pool of tokens whose cumulative probability exceeds `p`, a hyperparameter which we set to 0.9. To avoid selecting words with extremely low probabilities for the sensical sentence we initially limit the pool size to at most 50 words. After the "/" token is generated we increase the pool size to 100 in order to increase the variability of possible tokens for the non-sensical sentence. The mentioned hyperparameters were selected based on the quality of generated sentences, similarly to the procedure mentioned in Section 5.2.

Preprocessing. The preprocessing phase can be further broken down into 4 steps: (1) elimination of texts that do not match the structure of a valid sentence pair, (2) removing the texts which consist of the same sentences, (3) removing the pairs which are already a part of the ComVE training dataset, and (4) extracting the sentence pairs from the texts. In the first preprocessing step we remove all the texts that contain three or more sentences, texts that contain sentences with characters other than numbers, letters, and punctuation, or with two or fewer tokens. To tokenize the sentence and obtain the number of tokens we use GPT-2_{LARGE} tokenizer.

Selection. To perform selection of the generated pairs we leverage GPT-2_{LARGE} model. However, in this phase, we use it without fine-tuning. Firstly, we use lm-scorer[6] to assign scores to all generated sentences. The sentence score is calculated by aggregating token probabilities computed by GPT-2_{LARGE}. We consider four aggregation methods: arithmetic mean, geometric mean, product, and harmonic mean. In further research we use only the geometric mean and product, producing two datasets which we denote by GPT-2-G, and GPT-2-P, respectively. The aggregation methods are selected based on the performance of the method on the validation set. To calculate the performance of aggregation methods we firstly calculate the sentence scores using GPT-2_{LARGE} that employs a certain method. We then assign the labels to match the index of a sentence with a lower score. To measure the performance we use the accuracy score. Comparison of the performances is shown in Table 2. The product scoring achieved the best accuracy. This may be attributed to the fact that non-sensical sentences are slightly longer in all the datasets, as shown in Table 1. Since the token probabilities are in the interval $[0, 1]$, their product decreases with respect to sentence length. Therefore, sentences with fewer tokens are scored higher than longer sentences. On the contrary, other methods calculate some form of mean values, which leads to equal treatment of the sentences, regardless of their length.

Finally, we select 10k sentence pairs ranked by the highest difference between the scores assigned to a sensical and a non-sensical sentence. However, we evaluate only the models trained on top 5k artificially generated pairs concatenated to the original data since they achieve the best results on the validation set in comparison to using the top 2.5k or 10k artificially generated pairs. After the selection is performed, we randomly shuffle the sentences in each pair and assign the labels accordingly.

Mean word counts in each of the datasets are shown in Table 3. As expected, due to the nature of the aggregation methods, GPT-2-P contains shorter sentences on average in comparison to GPT-2-G. Figure 2 shows the trends in mean word count in sentence pairs with respect to the number of selected sentences.

[4]https://nlpaug.readthedocs.io/en/latest/augmenter/word/context_word_embs.html

[5]The code for fine-tuning and generation phases is the adaptation of https://github.com/prakhar21/TextAugmentation-GPT2. Due to resource limitations we were unable to perform hyperparameter optimization for fine-tuning so we used the hyperparameters proposed by the author.

[6]https://github.com/simonepri/lm-scorer

Table 4: The accuracies (%) of the models based on different encoders trained on the original dataset and the datasets generated by employing different text augmentation techniques. The amount of training examples in each dataset is shown in the brackets. #θ denotes the number of parameters of each encoder. In the "Ensemble" column we report the results of the voting ensemble of the models in the corresponding row. The best results in each row are in bold.

Model	#θ	Original (10k)	GPT-2-P (15k)	GPT-2-G (15k)	BT (18k)	CWE (20k)	Ensemble
BiLSTM	3.6M	61.2	63.4	62.5	62.9	61.4	**64.6**
ALBERT	11M	82.8	81.7	80.6	82.4	82.3	**83.8**
DistilBERT	110M	84.8	86.1	85.7	84.0	85.3	**86.4**
DistilRoBERTa	82M	85.7	85.2	85.9	85.6	86.2	**87.0**
ELECTRA$_{\text{SMALL}}$	13M	89.0	89.6	89.4	89.1	89.0	**90.1**
ELECTRA$_{\text{BASE}}$	110M	95.0	93.7	94.6	94.7	94.6	**95.6**

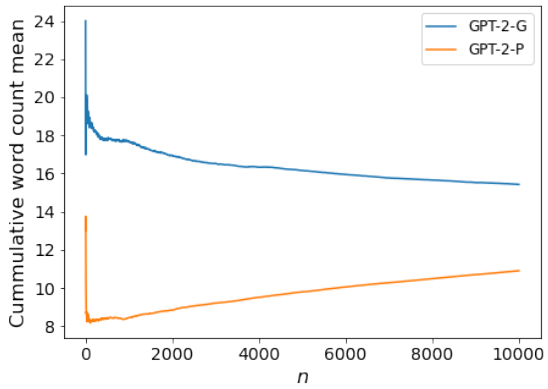


Figure 2: Cumulative means of word counts in top n sentence pairs by with the highest difference between a sensical and a non-sensical sentence in a pair. For example, if n is 100 the point in the graph corresponds to the mean word count in the top 100 pairs.

6. Results

Our main results are presented in Table 4. We evaluated our models on the test set and used the accuracy score as the metric as proposed by the authors of the ComVE task. We refer to models trained on the original dataset as baselines. For statistical significance testing, we employ the one-tailed permutation test with 10000 rounds at $\alpha = 0.05$.

The results show that the models perform similarly when trained on different augmented datasets. There is no notable difference in the performance of the models trained on GPT-2-P and GPT-2-G datasets despite the significant difference in the average sentence length. Nevertheless, the results indicate performance gain when the predictions of the models are aggregated by the voting ensemble. The results of the permutation test show that the ensemble of BiLSTM-based models significantly outperforms the corresponding baseline ($p = 0.045$). However, there is no significant difference in performance between ensembles of models based on ALBERT ($p = 0.246$), DistilBERT ($p = 0.145$) DistilRoBERTa ($p = 0.178$), ELECTRA$_{\text{SMALL}}$ ($p = 0.181$), and ELECTRA$_{\text{BASE}}$ ($p = 0.238$) and their respective baselines. Nevertheless, the performance gain, although not significant when it comes to the transformer-based models, suggests that text augmentation contributes to the diversity in the predictions of the models, which leads to better performance of the ensembles.

Arguably the best results are achieved by models that use ELECTRA as the encoder. ELECTRA$_{\text{SMALL}}$ ensemble significantly outperforms both DistilBERT ($p = 0.005$) and DistilRoBERTa ($p = 0.003$) ensembles despite utilizing notably less parameters. ELECTRA$_{\text{BASE}}$-ensemble performs significantly better than the other models we consider ($p < 10^{-4}$) and its accuracy is comparable to the current state-of-the-art on this task (97.0 %) achieved by Zhang et al. (2020).

7. Conclusion

We show that text augmentation techniques do not significantly affect the performance of transformer-based common sense validation systems. On the contrary, we show that the ensemble of BiLSTM-based models trained on the original dataset and the augmented datasets significantly outperforms the corresponding baseline. Our best model, an ensemble of models based on ELECTRA$_{\text{BASE}}$, achieves 95.6 % accuracy, which is comparable to state-of-the-art.

In future work, we would like to experiment with the proposed GPT-2-based text augmentation technique on less knowledge-intensive tasks. Furthermore, we believe we can improve the proposed technique by utilizing different sampling methods for text generation in order to increase the variability of generated texts. We would also like to experiment with the technique in a low-data regime.

References

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *CoRR*, abs/1904.09751.

Josef Jon, Martin Fajcik, Martin Docekal, and Pavel Smrz. 2020. BUT-FIT at SemEval-2020 task 4: Multilingual commonsense. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 374–390, Barcelona

(online), December. International Committee for Computational Linguistics.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China, December. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations.

H. Liu and P. Singh. 2004. Conceptnet — a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226, October.

Shilei Liu, Yu Guo, BoChao Li, and Feiliang Ren. 2020. LMVE at SemEval-2020 task 4: Commonsense validation and explanation using pretraining language model. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 562–568, Barcelona (online), December. International Committee for Computational Linguistics.

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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz 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*, volume 30. Curran Associates, Inc.

Cunxiang Wang, Shuailong Liang, Yue Zhang, Xiaonan Li, and Tian Gao. 2019. Does it make sense? and why? a pilot study for sense making and explanation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4020–4026, Florence, Italy, July. Association for Computational Linguistics.

Cunxiang Wang, Shuailong Liang, Yili Jin, Yilong Wang, Xiaodan Zhu, and Yue Zhang. 2020. SemEval-2020 task 4: Commonsense validation and explanation. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 307–321, Barcelona (online), December. International Committee for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October. Association for Computational Linguistics.

Yice Zhang, Jiaxuan Lin, Yang Fan, Peng Jin, Yuanchao Liu, and Bingquan Liu. 2020. CN-HIT-IT.NLP at SemEval-2020 task 4: Enhanced language representation with multiple knowledge triples. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 494–500, Barcelona (online), December. International Committee for Computational Linguistics.

Qian Zhao, Siyu Tao, Jie Zhou, Linlin Wang, Xin Lin, and Liang He. 2020. ECNU-SenseMaker at SemEval-2020 task 4: Leveraging heterogeneous knowledge resources for commonsense validation and explanation. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 401–410, Barcelona (online), December. International Committee for Computational Linguistics.

End-to-end vs Handcrafted Features for Author Profiling. Why not Both?

Jakob Domislović, Vjeran Grozdanić, Patrik Mesec

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{jakob.domislovic, vjeran.grozdanic, patrik.mesec}@fer.hr

Abstract

Author profiling is the problem of predicting an author's demographic information based on their texts. Similar to other tasks in natural language processing, this is a tough task for computers. We, as humans, can manually extract features from the text that can be useful for author profiling. Today, we are witnessing a huge leap forward in making attention-based models such as Bidirectional Encoder Representations from Transformers (BERT) that can automatically extract important parts of the text for the task. In this paper, we explore if there is any improvement in author profiling if we combine our own handcrafted features with BERT.

1. Introduction

Problem in which we want to predict authors age or gender from text is called author profiling. Authors often unconsciously make decisions on how to choose and combine words and this is why it is possible to construct a system that can automatically detect the author's age and gender. Being able to automatically profile authors has great usage in areas such as marketing, security, forensics, etc.

In this paper, we have focused on predicting the author's gender (male or female) and in which age group the author belongs. Predicting the exact age of the author is an overly demanding problem, but it can be made easier by predicting in which age range (e.g. 13-17 or 23-27) would the author's age fall.

A lot of work has been devoted to designing a set of features that can be used to build systems that can predict the author's age group or gender, but that task isn't the focus of this paper. Our approach was a little different. The motivation behind this paper is to show that, both in life and when designing a model, one should not be exclusive, but should combine different approaches.

The most interpretable models are the ones that use only the handcrafted features and easy to interpret machine learning algorithms (e.g. Support-Vector Machines or Decision Trees), but models like this are not expected to perform very well, unless a lot of attention is paid towards designing and extracting the features. At the other end, we are witnessing increased use of end-to-end models, such as Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) which has above average performance on profiling tasks.

By looking at two possible approaches, the one with handcrafted features and BERT being the other one, one question arises. Can we improve the performance of the system by combining these two approaches? The answer to that question can be found in the following chapters.

2. Related Work

Several works have been carried out on author profiling, especially on datasets such as Facebook, Twitter or Reddit corpora. Most of the previous work focused on either handcrafted features (Agrawal and Gonçalves, 2016) or automatic feature extraction with end-to-end model (Zhang and Abdul-Mageed, 2019). The former uses classifiers such as Bayesian Logistic Regression or Linear SVM, while the latter uses end-to-end model BERT. Comparing the results, although the models were not trained or tested on the same datasets, the end-to-end model wins over the handcrafted features. Our idea of combining handcrafted features with a neural network-based model was already proposed in (Shahmohammadi et al., 2021) and their paraphrase detection task achieved great performance.

3. Dataset

The dataset used for training and testing of the models is The Blog Authorship Corpus[1] which contains 681 288 posts from 19 320 bloggers, all written in English. This dataset contains text from blogs written on or before 2004, with each blog being the work of a single user.

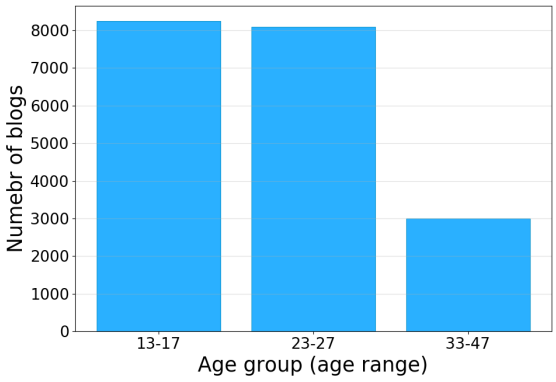


Figure 1: Age group details

To make age classification easier, we have grouped individual years of the authors into three groups whose distribution can be seen in Figure 1. The three age groups are: $13-17$ (ages between 13 and below 17), $23-27$ (ages between 23 and below 27) and $33-47$ (ages between 33 and below 47). By doing this grouping, our model has the task to solve a three-class classification problem.

[1] https://www.kaggle.com/rtatman/blog-authorship-corpus

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 33–36, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

The other classification problem is gender prediction. This dataset has come with genders already labeled and in it there are 49% female labels and 51% male labels.

4. Model

4.1. Architecture

BERT is a pre-trained transformer-based model published in 2018 by Google. Originally there were two English-language BERT models, the smaller one called "BASE" and the bigger one called "LARGE". Both models are pre-trained from unlabeled data extracted from the BooksCorpus and English Wikipedia. When the model was published, it achieved state-of-the-art results on many NLP tasks, such as question answering and general language understanding. Recently, many versions of BERT models were published, such as ALBERT, RoBERTa and DistilBERT. The goal of these recent models is mostly to reduce memory consumption (fewer parameters) and speed up the training process while preserving performance. The pre-trained BERT model can be easily fine-tuned for a more specific task, such as gender or age classification. Such fine-tuning (sometimes called transfer learning) includes adding layer(s) on top of the BERT model. Fine-tuning implies optimizing parameters of added layer(s). On the other hand, parameters of the BERT model don't have to be further optimized. Such a fine-tuned model is truly end-to-end. Each BERT model comes with its tokenizer which is used to tokenize raw input sentence/text/blog and to mask relevant tokens.

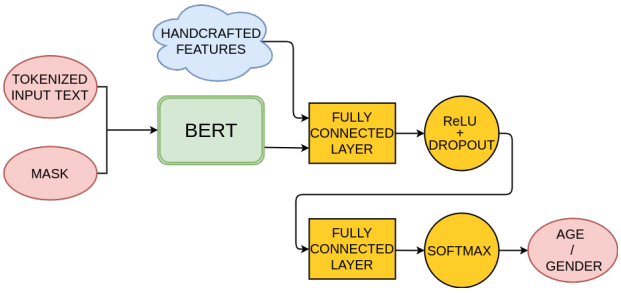


Figure 2: Model architecture

Due to hardware limitations, we used DistilBERT which has 40% fewer parameters and runs 60% faster than the "BASE" BERT model while preserving over 95% performance (Sanh et al., 2019). Due to memory limitations, the maximum sequence length that BERT accepts is 512. After tokenizing our raw input texts with the DistilBert tokenizer, we had to truncate/pad tokenized texts to overcome the maximum sequence length that BERT accepts problem. The mean length of all texts is 200, so we truncated/padded each sample to that length.

So far we have described the BERT model as an end-to-end model. In this paper, we introduce the idea of combining such an end-to-end model with handcrafted features to hopefully get the best of both paradigms. There are many ways and places where handcrafted features can be put in an already big and complex model as additional information. Before we show how we combine the BERT model with handcrafted features, let's describe additional layers that we added on top of the BERT model.

In our forward propagation, the model firstly computes the output of the BERT model. Using a part of that output, propagation continues through a fully connected layer (with the dimensionality of 256) followed by ReLU activation function and dropout with the probability of 0.1 which helps prevent overfitting. The final output is generated through a fully connected layer of dimensionality depending on the task (3 for age task and 2 for gender task) followed by a softmax activation function.

Handcrafted features were added to the input of the first fully connected layer of the model. We concatenated (horizontally stacked) part of the output of the BERT model with handcrafted features, so the dimensionality of the input size of the first fully connected layer is the sum of part of BERT output dimensionality and dimensionality of handcrafted features. The final scheme of model can be seen on Figure 2.

The output of our DistilBERT model has the following shape (batch size, sequence length = 200, hidden state dimensionality = 768). Following fully connected layer accepts input of size (batch size, hidden state dimensionality of BERT output + dimensionality of handcrafted features) so we had to deal with sequence length dimension from the BERT output. There are many options, such as averaging through sequence length dimension, but we decided to use only the first index of sequence length dimension (CLS token).

4.2. Handcrafted Features

We have used 14 following handcrafted features, these are the blog topic and the counts of sentences, characters, spaces, characters excluding spaces, duplicates, words, emojis, whole numbers, alphanumeric, non-alphanumeric, punctuations, stop words, noun phrase. The blog topic is one hot encoded feature which size is equal to the number of topic categories on dataset training split (including unknown). All features excluding the blog topic feature were evaluated and standardized before training the model. Handcrafted features of validation and test dataset split were standardized using means and variances of handcrafted features from training split.

4.3. Training

We split our dataset into three parts: training dataset (70%), validation dataset (15%) and test dataset (15%). We trained two models for each task. One model contained handcrafted features, and the other one did not. The used optimizer was AdamW with a 1e-4 learning rate and batch size 32. We optimized parameters of added layers and froze DistilBERT parameters.

A cross-entropy loss function was used for both tasks. We used weighted loss for the age classification task because of class imbalances, which can be seen in Figure 1. Due to hardware limitations, we managed to train each model for each task through 4 epochs on GPU. Training time for each model can be seen in Table 1. There is no big difference in training time with or without additional handcrafted features.

Table 1: Training time

Prediction Task	Time in minutes
Age without features	391
Age with features	393
Gender without features	396
Gender with features	397

5. Experiment and Results

Our assumption was that model which consists of both BERT and handcrafted features would outperform just the BERT model or just a simple ML model made from handcrafted features. We have conducted an experiment to empirically show if our assumption was correct.

The experiment has been made on three models: the decision tree with handcrafted features, the BERT model and the BERT model with the addition of handcrafted features (described in section 4.).

Each of the models has been first trained on gender prediction task and then, after the evaluation, each of the models has been trained on age category prediction task and evaluated. Models are trained on around 476 000 posts and then are validated on around 100 000 posts. In the end, each model is evaluated on the same test set which contains around 100 000 posts.

5.1. Gender Prediction Results

The results of gender prediction tasks can be seen in the Table 2, we use macro F_1 score. To show that there is a difference in the prediction labels between models, we have decided to use McNemar's test (McNemar, 1947) which is the test whose goal is to show that there is a difference in the marginal probabilities for each outcome. McNemar's test only works for two groups (models in our case) and two categories (*male* and *female* in our case). With that in mind, we have decided to only test if there is a difference between the predictions made by the BERT model and the BERT model with additional handcrafted features.

Table 2: Gender prediction results

Model	Accuracy	F_1 score
Decision tree	0.5051	0.3646
BERT	0.6699	0.6786
BERT + handcrafted features	**0.6996**	**0.6809**

After applying McNemar's test, we can reject H_0 hypothesis which states that there is no difference in the predictions of the models and conclude that there is a statistically significant difference between models. This means that predictions by the BERT model differ from the predictions by the BERT model with additional handcrafted features and if we look at the Table 2, we can also notice that the BERT model with additional handcrafted features has greater accuracy and macro F_1 score than other models.

5.2. Age Group Prediction Results

The results of age group prediction tasks can be seen in the Table 3, we use macro F_1 score. Unlike gender, for age we have three possible prediction categories (one for each age group). This means that we have to use the McNemar-Bowker(Krampe and Kuhnt, 2007) test to be able to statistically infer that predictions by the models differ. McNemar-Bowker is similar to McNemar's test, but with the difference that it works with multiple categories (not just two). Again, we are only testing if there is a significant difference between the predictions made by the BERT model and the BERT model with additional handcrafted features.

Table 3: Age prediction results

Model	Accuracy	F_1 score
Decision tree	0.4709	0.2173
BERT	0.62196	0.5934
BERT + handcrafted features	**0.6803**	**0.6620**

After applying the McNemar-Bowker test, we can again reject H_0 (p value is 0.0) hypothesis which states that there is no difference between the predictions of the models. This again means that there is a statistically significant difference between predictions made by the BERT model and the BERT model with additional handcrafted features. Also, if we look at the Table 3, we can notice that the BERT model with additional handcrafted features has higher accuracy and macro F_1 score than other models.

6. Conclusion

In this paper, we presented an overview of the author profiling problem which was addressed by training two different classifiers, one for the author's age and one for the author's gender. Our main goal was to see whether we get different results if we combine handcrafted features with end-to-end model BERT. Generally, gender classification was demonstrated to be an easier task than age classification, which is expected since there are only two classes for gender, while for the age we have multiple classes.

As shown in the paper, we did get more satisfying results by combining our handcrafted features with a pre-trained BERT model. Handcrafted features have especially improved our age classification task.

In future work we would like to test some other end-to-end model (e.g. XLNet) with handcrafted features and also we would like to design and extract more complicated features from posts in the hope of achieving better results.

References

Madhulika Agrawal and Teresa Gonçalves. 2016. Age and gender identification using stacking for classification notebook for pan at clef 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Anne Krampe and Sonja Kuhnt. 2007. Bowker's test for symmetry and modifications within the algebraic framework. *Computational statistics & data analysis*, 51(9):4124–4142.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Hassan Shahmohammadi, MirHossein Dezfoulian, and Muharram Mansoorizadeh. 2021. Paraphrase detection using lstm networks and handcrafted features. *Multimedia Tools and Applications*, 80(4):6479–6492.

Chiyu Zhang and Muhammad Abdul-Mageed. 2019. Bert-based arabic social media author profiling. *arXiv preprint arXiv:1909.04181*.

Can We Marry Machine Learning with Personality Psychology? In Theory, Yes.

Fran Jelenić, Marija Šokčević, Ana Vukasović

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{fran.jelenic,marija.sokcevic,ana.vukasovic}@fer.hr

Abstract

Personality assessment is an important part of understanding human nature. Progress in machine learning looks as a promising alternative to classic personality tests as well as an opportunity to further personality psychology theory. Traditionally personality tests need to be rigorously validated to be used as evaluation tools. If we expect machine learning personality assessment to become widely used it should be tested just as robustly. In this paper, we considered testing some aspects of construct validity such as reliability. Unlike previous work, our approach is based on basic machine learning models and a simple dataset containing solely raw text without any extra features.

1. Introduction

Personality psychology is a branch of psychology that studies personality and its variation among individuals. It aims to show how people are individually different due to psychological forces. Personality is defined as individual's differences in characteristic patterns of thinking, feeling, and behaving. Knowing someone's personality can benefit in more than one purpose: assessing theories, evaluating the effectiveness of therapy, diagnosing psychological problems, looking at changes in personality and screening job candidates.

Big Five personality assessment is one of the best known personality traits theories. It was developed in 1980s and identifies five factors: extraversion (outgoing/energetic vs. solitary/reserved), agreeableness (friendly/compassionate vs. critical/rational), openness to experience (inventive/curious vs. consistent/cautious), conscientiousness (efficient/organized vs. extravagant/careless), and neuroticism (sensitive/nervous vs. resilient/confident).

Most of the measured Big Five personality traits assessments rely on self-report questionnaires. A question is raised whether machine learning can be an alternative to classic personality questionnaires. Machine learning has been shown successful in predicting people's personality from text. However, most machine learning for personality assessment is focused on their performance, and less on the purpose of furthering the science behind personality theory. Some research has been done on this topic, but mostly on feature rich datasets.

Within this paper we try to show that even simple models and simple data can contribute to learning more about personality traits assessment.

2. Related Work

The field of personality assessment with machine learning received great attention in recent years. Most research has been done on data derived from social media platforms. With the increase of users on different social platforms and expanding the experience of their usage, more and more information is readily available. In addition to features such as age, gender, interests and posts available from users' social media profiles, even online surveys and questionnaires can be integrated within a social media platform and can then be easily connected (Kosinski et al., 2015). A great example of such machine learning for personality assessment with data from Facebook users is proposed in Park et al. (2015).

Data from social media is useful for its abundance of features and information, but not every dataset is like that. Another popular dataset for personality assessment is essays. It was first introduced in Pennebaker and King (1999) but has reached its peak in Majumder et al. (2017), where they use deep-learing based approach that is considered one of the best performing models for the dataset. As is expected, models that use social media datasets with more information have better performance.

Bleidorn and Hopwood (2018) provide broader context of fundamental principles of construct validation in machine learning personality assessment and provide recommendations for how to use machine learning to advance our understanding of personality. It reconciles psychological personality theory with machine learning practices. Most of the work trying to validate machine learning models and make them as robust as traditional personality tests has been done on social media datasets. We will try and implement same robust validity on a simpler dataset with less information.

3. Dataset

We used the labelled essay dataset from Pennebaker and King (1999). The dataset consists of 2467 entries whose inputs are stream-of-consciousness texts written by students. Each of the entries has five output labels for each of the Big Five personality traits - openness, conscientiousness, extraversion, agreeableness, and neuroticism.

It is known that each student has written multiple essays, and they had 20 minutes for writing one. However, the texts are anonymous and there is no other information than the essay text and output labels. This was a good match for our challenge to implement robust validity on a modest dataset.

4. Reliability

Reporting reliability statistics is standard in psychology research. We say the model is reliable if it shows consistency

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 37–40, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

Table 1: Mean F1 score of Big Five traits for different models on 10-fold cross validation

Model	F1 score				
	O	C	E	A	N
LR	**0.649**	0.594	0.596	0.628	0.605
SVM (RBF)	0.637	**0.604**	**0.642**	**0.667**	0.596
SVM (linear)	0.628	0.592	0.609	0.625	**0.606**

Table 2: Comparing our mixed model to the stratified dummy guesser based on mean F1 score for Big Five traits

Model	F1 score				
	O	C	E	A	N
Dummy model	0.495	0.506	0.522	0.546	0.492
Mixed model	0.643	0.595	0.635	0.664	0.588

of test scores across time, raters or content. Since we do not have data like time or raters, but only the text itself, we have devised a way in which we can still test reliability. First we selected the model which we will later use for the experiment.

4.1. Model Selection

We experimented with several different models in order to find the optimal ones for each trait. This was a crucial step for our experiment. We considered L2-regularized Logistic Regression and Support Vector Classifier with two different kernel functions (linear and RBF).

To prepare the dataset we decided to use *spaCy*[1] open-source library for Natural Language Processing (NLP). First we did the tokenization and lower-casing of the essays. We then removed stop words and punctuation. To vectorize the essays *spaCy* averages their token vectors.

Table 1 conveys the mean F1 scores of the models on 10-fold cross validation. We note that the highest F1 score for each personality trait was achieved with different model. As can be seen in the table, Logistic Regression outperformed the SVM models for the openness trait. In other traits, SVM did better and scored higher with linear kernel function for neuroticism, while the RBF kernel function had better results for conscientiousness, extraversion and agreeableness. Thus, we decided we would use the combination of best performing models and combine it into one mixed model.

As a baseline to evaluate our personality trait prediction model we used a simple dummy classifier that generates predictions by respecting the training set's class distribution. The mixed model was then evaluated against the stratified dummy model. The results are shown in Table 2. We conclude that the F1 score of our mixed model is significantly better than the baseline using t-test on a 10-fold cross validation with p-value < 0.05.

4.2. Experiment

The classic example of reliability testing in machine learning models for personality assessment is illustrated in Park et al. (2015). In their traditional test-retest approach they test the reliability by partitioning the data per person into six month subsets and then reevaluating personality traits. Their dataset is acquired from social media meaning they have a plethora of features available including time, age, gender, interests, etc.

[1] https://spacy.io/

Our stream-of-consciousness essays dataset lacks features needed for a full scale reliability test. Therefore, we needed to come up with a new approach for reliability testing. Since the essay entries are anonymous, the only thing we know is that one essay was written by one person. The idea we came up with was to divide the essays from the test set into two separate essays to simulate the split-half method. This way we created two artificial essays written in the style of the same person. To do so, sentences from each essay were randomly distributed between two essays. Newly created artificial essays consist of half of the sentences of the original essay. We expect these two would be classified with same personality traits.

5. Principles of Inference

In psychology it is important to know the underlying theory of the construct that the test is designed to measure (Haynes et al., 1995). Content validity emphasizes this idea. Most existing models are focused on optimizing their F1 score and accuracy while neglecting content validity or any theoretical background for that matter. A step in the right direction would be to combine machine learning for personality assessment and underlying psychological theory.

5.1. Model Selection

Models we used for this purpose are Logistic Regression and Decision Tree. We regularized these models with L2-regularization for Logistic Regression and by limiting maximum depth for Decision Tree model. This was done in order to turn off unimportant features.

To prepare the data for this experiment, we did similar preprocessing as in model selection for reliability testing, except this time we did not use built-in vectors. We opted for one-hot encoded ngrams which included both unigrams and bigrams. One-hot is a vector among which the legal combinations of values are only those with a single dimension with the value of 1 and all the others with the value 0. To get the text representation all of its unique ngrams' one-hot vectors were summed up. This way each ngram was a feature.

5.2. Experiment

For this experiment the idea was to see if different models have the same way of making their conclusions. In other words, we tested two models to get the features they attach the most importance to. We were also curious to see which features would be rated as most significant ones for each trait.

Logistic Regression can be somewhat interpretive when using one-hot vectors because model's weights give meaning to each ngram. Decision Trees are simple to understand and interpret, and trees can be visualized. The most important ngrams are easily extracted from the tree.

For measure of important features in Decision Tree we used mean decrease impurity. Since the mean decrease impurity has values between 0 and 1, and Logistic Regression can have both positive and negative weights (although due to L2-regularization also preferring smaller absolute values) we calculated pearson correlation coefficient between absolute values of Logistic Regression weight vector and Decision Tree feature importance.

6. Results

In this section we present and attempt to explain the results of our conducted experiments.

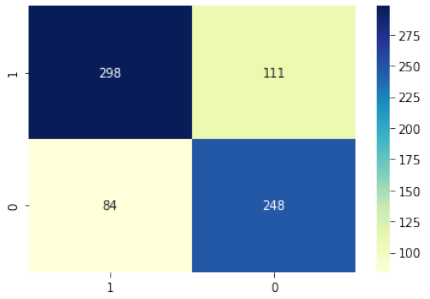
Reliability Our experiment's results, shown in form of confusion matrices for each trait in Figure 1, prove that the model is reliable. Classifications of two artificial "essays" created from the original text for each of the personalities is significantly dependent ($p < 0.05$, using χ^2 test). The calculated phi-correlation coefficients range from 0.389 for neuroticism to 0.544 for extraversion. This is to be expected because two essays were derived from the same essay written by the same person.

Principles of Inference In the second experiment we explored the difference in relevance that the models give to the same features. We expected significant correlations of importance two models give to their features because we hope they have the same underlying principles of inference. For each trait we calculated pearson correlation between feature importance of Logistic Regression and Decision Tree classifier. We got statistically significant correlation coefficients ($p < 0.05$) ranging from 0.120 for agreeableness to 0.150 for conscientiousness.

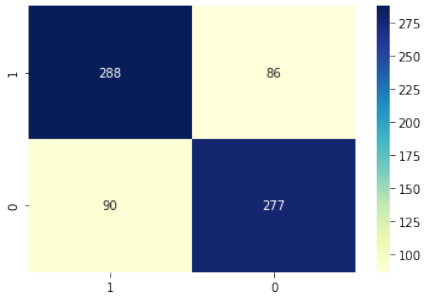
Furthermore, we extracted ngrams most important for classification. Since Logistic Regression and Decision Tree show significant correlation we used Logistic Regression weights with highest magnitudes to do so. Table 3 shows the most important ngrams for each trait. Some obtained ngrams make sense like "love" and "college student" for extraversion or "ok" for agreeableness, while other ngrams like "pop" for intraversion or "wednesday" for negligence (opposite of conscientiousness) are not quite obvious. It is difficult to use these words without context to determine the sentiment in which the author used those words.

7. Conclusion

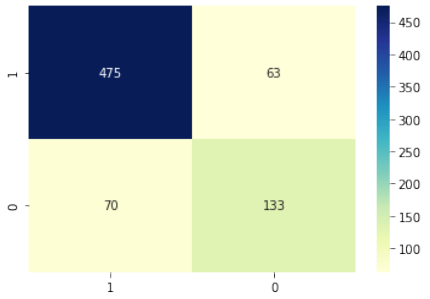
In this paper we highlight the need for better inclusion of psychology theory in machine learning personality assessment, even on the simplest of data. Traditional personality test based on self-assessment might have a bias due to subjectivity. In contrast, machine learning models do not suffer from such bias, and could potentially be used as an alternative or alongside traditional personality tests. Hence, machine learning should be validated as robustly as their counterparts. This could be applied in several fields, for instance job applicants could write their own stream-of-consciousness essay that could be evaluated alongside



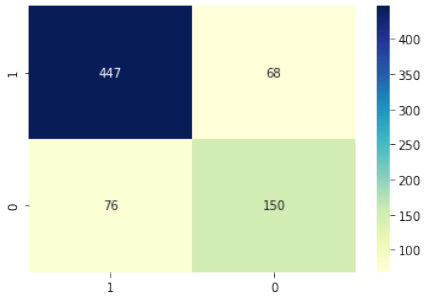
(a) openness



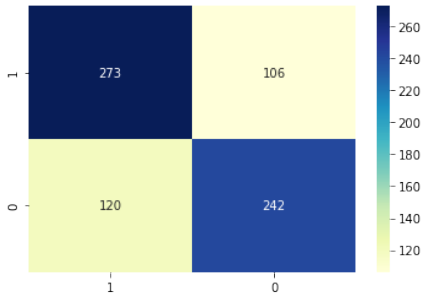
(b) conscientiousness



(c) extraversion



(d) agreeableness



(e) neuroticism

Figure 1: Distributions of classifications of pairs of artificial essays for each trait.

Table 3: Unigrams and bigrams that contribute most to the polarity of Big Five personality traits. "+" means it contributes to being classified positive polarity, and "-" means it contributes to being classified negative polarity.

Big Five personality trait	Polarity	
	+	-
Openness	cat, member, extreme	forever, compare, bathroom
Conscientiousness	explain, tick, lucky	got, wednesday, chance
Extraversion	love, schedule, college student	machine, pop, memory
Agreeableness	ok, feeling, sink	depressed, cafeteria, stupid
Neuroticism	sex, stupid, survive	shoot, ex, beat

their self-assessment personality test. We incorporated reliability (an aspect of construct validity) and principles of inference within our experiments and show it is possible to include psychology theory even with simple models.

Important part of furthering the personality theory through machine learning is understanding how models come to their conclusions. It is hard to understand the authors opinions and personality just on features as simple as words they use without understanding the whole context. Therefore, we suggest future work includes sentiment analysis for easier interpretability of used words and contexts they are used in.

References

Wiebke Bleidorn and Christopher James Hopwood. 2018. Using machine learning to advance personality assessment and theory. *Personality and Social Psychology Review*, 23(2):190–203. PMID: 29792115.

Stephen Haynes, David Richard, and Edward Kubany. 1995. Content validity in psychological assessment: A functional approach to concepts and methods. *Psychological Assessment*, 7:238–247, 09.

Michal Kosinski, Sandra Matz, Samuel Gosling, Vesselin Popov, and David Stillwell. 2015. Facebook as a research tool for the social sciences. *The American psychologist*, 70:543–556, 09.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, and Erik Cambria. 2017. Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, 32(2):74–79.

Gregory Park, H. Schwartz, Johannes Eichstaedt, Margaret Kern, Michal Kosinski, David Stillwell, Lyle Ungar, and Martin Seligman. 2015. Automatic personality assessment through social media language. *Journal of personality and social psychology*, 108, 11.

James W. Pennebaker and Laura A. King. 1999. Linguistic styles: Language use as an individual difference. *Journal of Personality and Social Psychology*, 77(6):1296–1312.

Linguistic Features Impact in Stress Analysis

Domagoj Knežević, Josipa Tomić, Nikola Vugdelija

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{domagoj.knezevic,josipa.tomic,nikola.vugdelija}@fer.hr

Abstract

In recent years, researchers of human psychology have gained a valuable ally, the Internet, specifically social media. Increasing number of posts on social media serves as a never-ending stream of data upon which different aspects of psychology can be explored. One of those aspects is stress and its influence on human behaviour. As such, its identification can be proven very useful in various science fields. In this work, along with proposing a new stress classification model using a combination of transformers, logistic regression and support vector machine, we mainly focus on linguistic features and their effect on system performance using various combinations of the aforementioned features. Our model shows performance improvement compared with related experiments.

1. Introduction

Stress occurs daily in various spheres of life. In some cases, it can have a positive effect on people who experience it, but it is mostly a negative phenomenon that worsens the way of life. For this reason, its detection is crucial in studying, understanding, and relieving stress. In recent years, there has been an increase in the use of social networks, and thus the number of posts on the Internet in which we can investigate stress, also it has become easier to detect it than before the emergence of social networks. Since 2020 was marked by the COVID-19 pandemic, a lot of research has been done to prove how much the level of stress in certain countries and in certain strata of society increased during the pandemic period (Taylor et al., 2020).

According to Mental Health Foundation[1], stress can be defined as a feeling of being overwhelmed or unable to cope with mental or emotional pressure. It is our body's response to pressure and every individual reacts differently to its onset.

Regarding the detection of stress, research on physical reactions to stress has often been performed through measurements of brain activity, also known as EEG (electroencephalogram) (Al-Shargie et al., 2016), or speech of the affected person (Zuo et al., 2012), as well as the levels of certain hormones in saliva (Allen et al., 2014). In recent years, stress detection work has appeared on written, textual data, which is discussed in more detail in section 2. Our work can be conceived as two subtasks. The first is the introduction of a new model, i.e. the combination of existing models, in order to achieve competitive results. The second task, and in fact the main subtask, aims to determine which features affect the performance of the system most favorably, in order to gain insight about those features. What differs our work from similar ones is the fact that we focus on the influence of linguistic features both for the purpose of determining, and for separating the useful from the less useful features for the performance of the system itself.

In the following sections, we compare our work with the most similar works done, and show what differences adorn us. We present and describe the dataset on which the analysis was performed, describe the baselines with which we compared our work, and describe our approach to the tasks. It remains to show the results of our work and the conclusions we have drawn from it.

[1] https://www.mentalhealth.org.uk/a-to-z/s/stress/

2. Related Work

Stress detection in textual data only came to life with the advent of social media, as researchers had a larger set of data they could work on. Often these text entries are shorter, like Twitter or microblogs, but also Facebook. Lin et al. (2017) use CNN (Convolutional Neural Network) to detect stress on microblogs. Winata et al. (2018) use LSTM (Long Short Term Memory) in order to detect stress in posts from Twitter, but also from speech. Guntuku et al. (2018) have a questionnaire after which they analyze adequate candidates' posts on Facebook and Twitter.

Turcan and McKeown (2019) deal with the construction of a dataset called Dreaddit used for stress detection, part of which we use in our work, where it is important to note that the dataset is constructed from posts from the social network Reddit, which are generally slightly longer than posts from the aforementioned social networks. Also, they analyze the complexity and diversity of the data and characteristics of each category from the dataset. Given the above, we felt that deepening our knowledge of the effect of linguistic features used could prove useful in future research as it would show what to focus on when dealing with linguistic features and their usage in stress detection tasks.

3. Experimental Setup

In this section we describe dataset that is used, baselines for the first subtask and our method for tackling the second subtask.

3.1. Dataset

As already mentioned, the dataset used is called Dreaddit, which was created for research purposes (Turcan and McKeown, 2019). It is made up of a total of 3.5K posts from Reddit of which 80 percent is accounted for by the training set and the rest by testing set. Posts from 5 different domains were used, respectively abuse, anxiety, financial, PTSD, social, i.e. categories that were recognized as

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 41–43, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

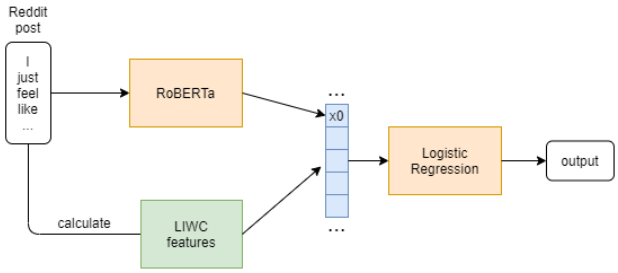


Figure 1: Architecture of the model used.

the best candidates to find posts with evidently expressed stress. In addition, LIWC features *(Linguistic Inquiry and Word Count)* have been published for each post, which is the focus of this research. LIWC[2] features highlight language features derived directly from the text and relate to various areas, such as the use of different pronouns, adverbs, prepositions. It also counts the percentage of words that reflect different emotions, thinking styles, social concerns, and even parts of speech. As such it can pose as a valuable resource in analyzing various emotional states. If interested, reader can get more information about this dataset in research from Turcan and McKeown (2019).

3.2. Baselines

As our baselines, we selected a logistic regression model that used pre-trained Google Word2Vec embeddings for text representation alongside random LIWC features acquired from the dataset. We also used an SVM with a linear kernel that uses the same features as the logistic regression model described previously for broader comparison. As another baseline, we applied RoBERTa directly for our stress detection task, while also fine-tuning it.

3.3. Our Approach

An experiment was conducted in which we fine-tuned two transformers: BERT and RoBERTa. Both of those are bidirectional transformers pretrained using a combination of masked language modeling objective and next sentence prediction on a large corpus. We handed over Reddit posts from the whole data set to the transformers as inputs. After combining different parameters and the number of epochs and comparing the obtained results for both transformers, the RoBERTa transformer proved to be better and it was selected as the first layer of our model.

After the first layer was selected, we performed a logistic regression to which we passed all the LIWC features contained in the data set as input. In order to boost our model, we have added a RoBERTa transformer prediction to each element of the set as a feature. Our model architecture can be seen in Figure 1.

Second task was to determine which combinations of features have the greatest effect on our system performance. We first tried individual features and from the best combined them in different manners in order to find the best combinations. Results of our finding are explained in next section.

[2] http://liwc.wpengine.com/

4. Results

After running dataset posts through fine-tuned RoBERTa transformer, we used the transformer's output as input to the logistic regression model together with all the LIWC features individually in order to find the features that help to produce the best possible F1 score. We selected 10 features with which the model achieved the best F1 score and created every possible subset from them. Then we repeated the procedure but with subsets and ranked them again by their F1 scores. The results are presented in the Table 1.

Work feature represents the count of words regarding work, adverb represents the number of common adverbs mentioned in the text, shehe corresponds to amount of verbs in 3rd person singular form, drives stand for the count of affiliation/achievement words, ipron represents the number of impersonal pronouns mentioned in the text, semic/period represent corresponds to count of semicolumns/periods in the text, and the swear feature represents the amount of swear words inside the text.

If we compare the results of our best performing subset of features with the results of each of our baselines as well as the results of vanilla RoBERTa transformer we can see that our model outperforms these models in every metric except for recall. We can also see that the performance of the SVM model is identical to the performance of RoBERTa transformer. The results are displayed in the Table 2.

5. Conclusion

In this paper, we have proposed our solution for classifying Reddit posts based on the appearance of the signs of negative stress in the post. We have shown which of the features from the LIWC dictionary help the most with the classification, and proposed a model that uses the RoBERTa transformer to achieve better classification than what was achieved in the related work.

References

Fares Al-Shargie, M. Kiguchi, Nasreen Badruddin, Sarat Dass, Ahmad Fadzil Mohd Hani, and Tong Boon Tang. 2016. Mental stress assessment using simultaneous measurement of eeg and fnirs. 7:3882–3898, 10.

Andrew P Allen, Paul J Kennedy, John F Cryan, Timothy G Dinan, and Gerard Clarke. 2014. Biological and psychological markers of stress in humans: focus on the trier social stress test. *Neuroscience and biobehavioral reviews*, 38:94—124, January.

Sharath Chandra Guntuku, Anneke Buffone, Kokil Jaidka, Johannes Eichstaedt, and Lyle Ungar. 2018. Understanding and measuring psychological stress using social media. 11.

Huijie Lin, Jia Jia, Jiezhong Qiu, Yongfeng Zhang, Guangyao Shen, Lexing Xie, Jie Tang, Ling Feng, and Tat-Seng Chua. 2017. Detecting stress based on social interactions in social networks. *IEEE Transactions on Knowledge and Data Engineering*, PP:1–1, 03.

Steven Taylor, Caeleigh Landry, Michelle Paluszek, Thomas Fergus, Dean Mckay, and Gordon Asmundson. 2020. Development and initial validation of the covid

Table 1: Subsets that combined with RoBERTa transformer achieve the best F1 score.

Features used	**Accuracy**	**F1 score**	**Recall**	**Precision**
(Work, Adverb, Shehe, Drives, Ipron, Period, Swear)	0.82657	0.83598	0.85637	0.81654
(Work, Adverb, Drives, Ipron, Max Pleasantness, Period, Swear)	0.82657	0.83598	0.85637	0.81654
(Work, Adverb, Shehe, Drives, Ipron, SemiC, Period, Swear)	0.82657	0.83598	0.85637	0.81654
(Work, Adverb, Drives, Ipron, Max Pleasantness, SemiC, Period, Swear)	0.82657	0.83598	0.85637	0.81654
(Work, Adverb, Shehe, Drives, Ipron, Max Pleasantness, Period, Swear)	0.82657	0.83598	0.85637	0.81654
(Work, Adverb, Shehe, Drives, Ipron, Max Pleasantness, SemiC, Period, Swear)	0.82657	0.83598	0.85637	0.81654
(Drives, Ipron, Max Pleasantness, SemiC, Period, Swear)	0.82657	0.83554	0.85366	0.81818
(Work, Adverb, Shehe, Ipron, SemiC, Swear)	0.82657	0.83554	0.85366	0.81818
(Work, Adverb, Shehe, Drives, Ipron, Swear)	0.82657	0.83554	0.85366	0.81818
(Work, Adverb, Shehe, Drives, Ipron, SemiC, Swear)	0.82657	0.83554	0.85366	0.81818

Table 2: Comparison between the final model and the baselines

Model	**Accuracy**	**F1 score**	**Recall**	**Precision**
RoBERTa + Logistic regression (using most efficient features)	0.82657	0.83598	0.85637	0.81654
Vanilla RoBERTa	0.81678	0.83053	0.86991	0.79455
SVM (Pretrained embeddings + random LIWC features)	0.81678	0.83053	0.86991	0.79455
Logistic regression (Pretrained embeddings + random LIWC features)	0.81818	0.83117	0.86721	0.79800

stress scales. *Journal of Anxiety Disorders*, 72:102232, 05.

Elsbeth Turcan and Kathy McKeown. 2019. Dreaddit: A Reddit dataset for stress analysis in social media. In *Proceedings of the Tenth International Workshop on Health Text Mining and Information Analysis (LOUHI 2019)*, pages 97–107, Hong Kong, November. Association for Computational Linguistics.

Genta Winata, Onno Kampman, and Pascale Fung. 2018. Attention-based lstm for psychological stress detection from spoken language using distant supervision. pages 6204–6208, 04.

Xin Zuo, Tian Li, and Pascale Fung. 2012. A multilingual natural stress emotion database. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1174–1178, Istanbul, Turkey, May. European Language Resources Association (ELRA).

Cyberbullying, Show Yourself! Uncovering Bullying Traces With Modern Machine Learning and Deep Learning Models

Jakov Kruljac, Ena Rajković, Eugen Rudić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{jakov.kruljac, ena.rajkovic, eugen.rudic}@fer.hr

Abstract

While most of the people make good use of technology development and social media, some see an opportunity to victimize others. In recent years, bullying is not only present at the school playground and number of victims of cyberbullying is constantly growing. Social media platforms are the perfect example and offer a lot of data to analyze unwanted behaviour. In this paper, we collected around 16000 comments from Twitter platform. We built a model to detect bullying comments and did topic modeling on those comments to find out which words contribute and occur the most. We compare neural network model and support vector machine (SVM) model with added hand-crafted features formed with the help of topic modeling. Our results showed that features extracted with topic modeling can improve SVM classification, but those models still cannot beat scores of deep learning classification models.

1. Introduction

Nowadays social media sites have become one of the most popular ways to communicate and socialize.

Hundreds of millions of tweets are shared every day on Twitter social media platform. That's 6000 tweets per second. The openness of the platform offers vast amount of publicly accessible data, in which people often express their opinions and feelings.

A lot of social behaviour shifted to these sites and along with it a lot of unwanted behaviour such as bullying. This form of bullying is known as cyberbullying. Cyberbullying is defined as an intentionally aggressive behavior that involves an imbalance of power performed through electronic means (e.g., high-technology devices) (Hinduja and Patchin, 2007). More than every third student (age 12-17) is affected by it according to Hinduja and Patchin (2019). Students who experienced cyberbullying are almost twice as likely to attempt suicide compared to those who had not.

Because of the sheer amount of tweets it is practically not possible to hand-pick tweets containing some sort of bullying. This motivated us to introduce machine learning models for recognition of bullying comments.

Some work in this area has already been done, but not using state-of-art NLP methods and models like context based token embeddings and recurrent neural networks.

Related work will be presented in Section 2, followed by brief description of dataset. In Section 4 we will introduce and explain all parts of our model - SVM, neural network and topic modeling. Finally we discuss over the results in Section 5.

2. Related Work

Problem of cyberbullying exists for quite some time and there is a lot of systems whose goal is detection of cyberbullying on social media. These systems mostly rely on older methods of machine learning (SVMs, naive Bayes and logistic regression) and token representation (unigrams and bigrams). Xu et al. (2012) used basic categorization machine learning models on unigram and bigram POS-colored representation of tweets. Paper reported best results on SVM model with linear kernel with combination of unigram and bigram representations achieving precision P=0.76, recall R=0.79, and F-measure 0.77. Adding part-of-speech tagging didn't result in score improvement. Waseem and Hovy (2016) reported best results using logistic regression model on bi- to fourgram tokens with gender labels. This model scored F1=73.89. Paper from Ptaszynski et al. (2017) featured comparison table between a lot of cyberbullying classification models none of which used modern embeddings techniques e.g. GloVe and chars2vec.

Most of the existing research for topic modeling has been based on un- or weekly supervised techniques such as variations of Latent Dirichlet allocation (LDA). There's been work on comparison of the performance of LDA and the Dirichlet multinomial mixture model on short text (Mazarura and de Waal, 2016). Yin and Wang (2014) did Gibbs sampling algorithm for a Dirichlet Mixture Model. We used implementation based on their work.

3. Dataset

Dataset we used is gathered from Twitter platform, downloaded using the Twitter API. It consists of 16338 tweets written in English language, of which 5347 are labeled as bullying tweets. Two types are labeled for bullying tweets – sexism and racism. There is total number of 3377 sexist tweets and 1970 racist ones.

4. Models

In this paper we are approaching cyberbullying classification problem from two angles. One is trying to improve basic SVM model with modern embeddings adding the information of most frequent words in bullying comments. This approach relies on LDA topic modeling to gain insight on frequency and weight of those words. The other is end-to-end deep learning model based on LSTM architecture.

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 44–47, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

Figure 1: Wordclouds for racistic and sexist tweets

4.1. Topic Modeling

With new large amounts of data being created every day, it has become difficult to retrieve information we need. Topic modeling helps to discover patterns that occur in text collection and create annotations that can be used to search and summarize texts. It is a method of unsupervised classification (like clustering) that essentially finds a group of words from document collection that represents an abstract theme in that collection. One of the most popular topic modeling techniques is Latent Dirichlet allocation (LDA). It makes generative assumption that every document is a mixture of topics, and each topic is a mixture of words. Assumption is reasonable for longer texts, but short ones such as tweets deal with only one topic most of the time. In contrast to LDA, Gibbs Sampling algorithm for Dirichlet multinomial mixture (GSDMM) model assumes that each document is a member of only one topic. Implementation of LDA is taken from Gensim, together with slightly better Mallet's implementation. GSDMM model was implemented after Yin and Wang (2014). Parameters for the three models were chosen by random search.

Very important step before doing topic modeling on tweets is data preprocessing. In this case it meant removing links, email and newline characters, tokenization, removing stopwords (downloaded from NLTK) and lemmatization. Tokens were again concatenated to form clean tweet. To build a corpus for LDA, words were represented with their tf-idf score.

Topic modeling was performed in two experiments. Purpose of the first was to distinct bullying from not-bullying tweets. Whole dataset was used in this experiment. In the second one the goal was to create topics for types of bullying tweets. Since the data was already labeled with two types of bullying – sexism and racism, the number of topics to model was familiar. We created wordclouds (Figure 1) that display keywords for each type of bullying tweet.

By just observing words in two wordclouds, there are no obvious or common words that appear in both wordcloud for sexism labeled data and the one with racism label. In view of that fact, it can be expected that topic modeling won't give good results in the first experiment, because the model will have problems putting those two categories in a single bullying topic that opposes the non-bullying one. Therefore experiment I. will be expanded to model three classes – non-bullying, sexism, racism.

4.2. SVM

Semantic representations, namely GloVe and chars2vec, advanced NLP by a lot. In the learning from bullying in social media paper (Xu et al., 2012) those methods are overlooked and instead more traditional n-gram representation are used. We argue that, just by changing representation, models precision and F1 score could be drastically improved.

To test this hypothesis we rely on scoring comparison between our SVM classifier and one trained by Xu et al. (2012). Because of large difference between dataset sizes (n=16338 vs. n=1700) we sample only a subset of the dataset to avoid giving our model significant advantage. This subset is then split on train and test subset in 1500:200 ratio. This makes both models equal with dataset size. The model we are using is SVM model with linear kernel because it is the best scoring model for bullying trace categorization (Xu et al., 2012). For the sake of comparison we trained the same model on a different representation of tweets.

Each tweet is preprocessed with modified version of official GloVe preprocess script[1]. After preprocessing each word is embedded with 200 dimension GloVe twitter embeddings. Embedded tweet representation is calculated as a mean of GloVe embeddings. Summation results in a loss of information, but assumption is that this representation still preserves general context of the tweet.

In order to additionally improve our model performance we are going to expand the tweet representation with features gained from topic modeling. Features are counters of most prominent words found in bullying tweets. Those counts are multiplied by their corresponding frequencies found by topic modeling algorithm. After multiplication they are normalized and concatenated to the GloVe tweet representation.

4.3. Neural Network

In the last decade, with the development of Recurrent neural networks (RNNs), neural networks have been increasingly used for semantic text analysis. RNNs is a type of neural networks, which is used to capture time dynamic. The biggest difference from the other types is that RNNs does not assume that all inputs are independent and because of that output is a combination of input data and output data from previous input. Theoretically, RNN can capture long-term dependencies, but because of problem with exploding or vanishing gradient, RNN has poor results in that task. Long short-term memory (LSTM) is a type of RNN which is capable to capture long-term dependencies. LSTM unit has 3 gates: forget, input and output gate. In forget gate some data is discarded, in input gate memory is updated with new input data and in output gate memory and input data combine to create output.

4.3.1. Bi-directional LSTM

LSTM is designed to capture time dynamic, but only from past to present. In problems like deciding if text contains bullying trace or not, we want to have information from both directions. In that case, we use Bi-directional LSTM (BLSTM). BLSTM have two separate hidden states, one of hidden state is for capturing future and other capture past.

[1] https://www.kaggle.com/amackcrane/python-version-of-GloVe-twitter-preprocess-script

Table 1: Coherence scores comparison for topic modeling models

	LDA	Mallet's LDA	GSDMM
Experiment I.	0.123	0.699	0.345
Experiment I.*	0.235	0.716	0.409
Experiment II.	0.275	0.776	0.408

4.3.2. Network Architecture

Our network is a BLSTM neural network with fully connected layers as a decoder. For each word from our dataset first we use two pretrained models GloVe and chars2vec. Inputs in our BLSTM are combined word-level representation from GloVe and char-level representation from chars2vec. If a word from our dataset does not have GloVe representation, we use random initialization with uniform distribution. For the optimal model, we are using an Adam algorithm with learning rate 0.001 and batch size 32. We experimented with two GloVe models, one model every word represents with 300-dimensional vector and other model represent each word with 200-dimensional vector and is patterned on Twitter data. Also, we experiment with 300-dimensional and 100-dimensional chars representation. In our experiments if the model embedding matrix has more than 300 dimensions, fine-tuning was not allowed otherwise was allowed.

5. Results

5.1. Topic Modeling Results

Topic modeling is evaluated comparing coherence scores, calculated with the help of Gensim's coherence model. It measures the degree of semantic similarity between high scoring words in the topic. The results are shown in table 1. On the contrary to what we expected, GSDMM didn't give best results out of three models – Mallet LDA did. At least when comparing coherence scores. But if we look at generated topics for experiment II. and actual words that made them (in table 2), we can make sense of GSDMM topics easier than for topics generated from other two models. On the other hand, for experiment I., we can't clearly give labels to extracted topics, except maybe for GSDMM for modeling 3 topics (marked with * in table 1). This also confirms the beginning assumption that topic modeling on whole dataset would be better when modeling 3 topics (non-bullying, sexism, racism) rather than 2 (bullying and non-bullying).

5.2. SVM Results

Results of training SVM with GloVe based embeddings and with additional features modeled on insight gained with LDA topic modeling and those reported by Xu et al. (2012) are showed in table 3. Scores achieved by our GloVe and GloVe + topic feature representation are pretty close to the ones achieved by unigram + bigram based representation. We didn't manage to top the F1 score but GloVe + topic feature representation did show the best accuracy. Word frequencies, by which we modeled additional features, are found by GSDMM algorithm.

Table 2: Words derived from topic modeling

sexism	racism
sexist	muslim
woman	islam
girl	murder
kat	mohamme
man	isis
female	religion
call	woman
get	people
think	prophet
make	quran
go	year
say	jew
know	rape
see	kill
right	war
want	christian
feminist	terrorist

Table 3: F1 and accuracy scores of SVM models trained on combination of unigrams and bigrams, GloVe embeddings and GloVe embeddings with features gained from topic modeling

	1g+2g	GloVe	GloVe+Topic features
F1	0.77	0.651	0.670
Accuracy	0.76	0.754	0.776

5.3. Neural Network Results

Results of training BLSTM neural network are shown in two tables. In table 4 we show that model without char-level representation or word-level representation got worse results than model with both representations. In table 5 we showed results with different dimensions of word representation vectors. After all testing, we can see that neural network with GloVe and chars2vec representation where a word is represented by vector with 600 dimensions work best for recognition of bullying trace in comments on the internet.

Table 4: Result of BLSTM model with random vector representation, GloVe representation and GloVe + char2vec representation

	None	GloVe	GloVe+chars2vec
F1	0.65	0.70	0.71
Accuracy	0.76	0.79	0.82

Table 5: Result of BLSTM model with different vector dimension

	G200 + C100	G300 + C100	G300 + C300
F1	0.50	0.69	0.71
Accuracy	0.78	0.80	0.82

6. Conclusion

In this paper we tried to solve bullying comment classification problems using the state-of-art NLP methods. A lot of previous systems for bullying comments classification relied on machine learning classification models and n-gram representations. We wanted to contribute to this problem by including semantic and character level embeddings into standard SVM classifier and deep learning LSTM network. By exploring unsupervised topic modeling algorithms we gained insight on most prominent words in bullying tweets and used them as features in our SVM model. Of couple of models we tried, GSDMM model offered easiest interpretability of topics, but Mallet's LDA implementation gave the best coherence score. Even though adding handcrafted features to SVM improved results of the baseline SVM model (from F1=0.651 to F1=0.67), it still couldn't compete with deep neural network model (F1=0.70).

In the future work we plan to inspect deeper the importance of words extracted with topic modeling techniques and try increasing the number of topics to get maybe more fine-grained abstract themes. We believe that topic modeling could play an important role in feature engineering for machine learning models. Our neural network could be improved by adding new layers like convolutional ones. Also, because of a lot of hyperparameters and lack of time and resources, we didn't manage to do exhaustive grid-search to find optimal hyperparameters. We hope that could improve our results.

References

Sameer Hinduja and Justin W. Patchin. 2007. Offline consequences of online victimization. *Journal of School Violence*, 6(3):89–112.

Sameer Hinduja and Justin W. Patchin. 2019. Cyberbullying data.

Jocelyn Mazarura and Alta de Waal. 2016. A comparison of the performance of latent dirichlet allocation and the dirichlet multinomial mixture model on short text. pages 1–6, 11.

Michal Ptaszynski, J. Eronen, and Fumito Masui. 2017. Learning deep on cyberbullying is always better than brute force. In *LaCATODA@IJCAI*.

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, San Diego, California, June. Association for Computational Linguistics.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 656–666, Montréal, Canada, June. Association for Computational Linguistics.

Jianhua Yin and Jianyong Wang. 2014. A dirichlet multinomial mixture model-based approach for short text clustering. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 08.

Syntax vs Semantics: What Tells More About Your Personality?

Marko Lazarić, Laura Torić, Roman Yatsukha

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{marko.lazaric,laura.toric,roman.yatsukha}@fer.hr

Abstract

Personality traits are inherently subjective and hard-to-define categories, so classifying peoples' personalities is a challenging task even for people. On the other hand, natural language processing has made great progress in similarly subjective areas such as sentiment analysis. In this paper, we compare two approaches to classifying individual personality traits based on essays: syntactic and semantic. The syntactic approach uses the role of the word in the sentence, such as subject, object, verb, etc., while the semantic approach uses the meaning of the words themselves to classify the essays. We compare the two approaches on a dataset of 2467 student essays annotated with the big five personality traits and show that the semantic approach tends to perform better across the board.

1. Introduction

Every text contains a remarkable amount of information about the author. While that information might not be apparent to people, natural language processing has shown itself very adept at using that information to classify texts and their authors into certain categories.

An example of such a task is classifying the personality of the author based on their essay. Since a personality is hard-to-define, it is usually decomposed into several personality traits which are usually binary. The big five personality model is a popular personality trait model introduced in (Digman, 1990) and (Goldberg, 1993) which will be used in this paper. It uses the following five personality traits: extraversion, agreeableness, openness, conscientiousness and neuroticism.

In this paper, we experiment with two approaches to classifying the different personality traits: syntactic and semantic. The syntactic approach uses the role of the words in the sentence such as subject, object, etc. without knowing what the words mean. The semantic approach uses the meanings of the words, either through pretrained embeddings or the bag of words model. Our results show that semantic models tend to perform better than syntactic models for all personality traits.

2. Previous Work

(Park et al., 2014) used five separate statistical models to predict personality traits of Facebook users, using their posts and self-reported Big5 personality traits. They used both boolean encodings and relative frequencies of words and phrases, as well as topic extraction, as their features, which were then reduced using univariate feature selection, and randomized principal component analysis. Although we use different features for our predictions, we mirror the idea of using separate models for each personality trait rather than predicting them with a single model.

In contrast, (Pizzolli and Strapparava, 2019) used a single model on (Pennebaker and King, 2000) dataset with subpar results, although they used the essay dataset only for the purpose of having a baseline, before moving on to predict traits of various characters in William Shakespeare's Hamlet.

3. Dataset

The dataset used for this paper is a collection of student essays from (Pennebaker and King, 2000). The students' personality traits were assessed using the big five inventory (BFI) self-report questionnaire developed by (John et al., 1991). The dataset contains 2467 essays with a mostly balanced number of positive and negative examples for each class which is shown in table 1.

Table 1: The number of positive and negative examples for each personality trait.

Personality trait	Negative	Positive
Extraversion	1191 (48.28%)	1276 (51.72%)
Neuroticism	1234 (50.02%)	1233 (49.98%)
Agreeableness	1157 (46.90%)	1310 (53.10%)
Conscientiousness	1214 (49.21%)	1253 (50.79%)
Openness	1196 (48.48%)	1271 (51.52%)

Our goal is to correctly classify all of the big five personality traits based on the given essay.

4. Features

The features extracted from the essays were bag-of-words features and the number of occurrences of each part-of-speech and named entity tag in the essay. SpaCy[1] was used for lemmatization, part-of-speech tagging and named entity recognition.

To reduce the effect of the length of the essay on the features, alternative features were tried where each number was normalized by the sum of the part-of-speech or named entity tags to get the percentage of each tag or normalized by the number of sentences to get the mean number of each part-of-speech or named entity tag per sentence.

The previously explained features are shown in table 2 for the sentence "Well, right now I just woke up from a mid-day nap in Texas." SpaCy's abbreviations for individual tags are used to preserve space.

[1] https://spacy.io

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 48–51, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

Table 2: Number of occurrences of each part-of-speech and named entity tag detected by spaCy for "Well, right now I just woke up from a mid-day nap in Texas.".

Part-of-speech tags				Named entity tags	
RB	3	VBD	1	DATE	1
NN	3	RP	1	GPE	1
IN	2	DT	1		
UH	1	JJ	1		
,	1	NNP	1		
PRP	1	.	1		

For the semantic models, we used XLNet and its pretrained embeddings.

5. Models

Since the different personality traits are independent, a separate model was trained to classify each personality trait. To test the predictive power of syntactic features, SVMs were used, and XLNet was used as a representative semantic model. The individual models and their pipelines are further explained in the following subsections.

5.1. SVM

Support Vector Machines from scikit-learn[2] were trained to classify the essays for each personality trait. The SVMs used a radial basis function with the gamma parameter set to auto. Feature selection was done by selecting 10 features with the highest ANOVA F-value. After that, the selected features were standardized by subtracting the mean and scaling to unit variance and used to train the SVMs.

5.2. XLNet

XLNet is an auto-regressive language model introduced in (Yang et al., 2019). It uses transformers with recurrence to output the joint probability of a sequence of tokens. To calculate the probability, it uses all permutations of word tokens in a sentence, instead of just those to the left or right of the target token.

In this paper, we used a pretrained XLNet for sequence classification with the XLNet tokenizer from the transformers[3] package. The tokenizer takes the raw text and limits the input to 500 tokens so any essays longer than that are trimmed (about 70% of essays end up being trimmed). The network was trained for 5 epochs with a batch size of 5. Adam with the learning rate of 10^{-5} and 10^{-2} regularization was used to update the parameters.

6. Experiments

To test our syntax and semantic models we conducted three experiments described in this section.

Experiment 1. To test the performance of different models and features we decided to compare F1 and accuracy scores measured on the test set. For the syntax models, we used SVM with all possible combinations of part-of-speech and named entity tags and tested all of them. For the semantic models, we used XLNet and SVM with bag-of-words features.

Experiment 2. In order to test if some personality traits had different distributions of part-of-speech or named entity tags than others, we used a t-test, or more accurately, Welch's t-test. For a few selected most promising tags we tested the difference between every pair of traits and reported p values. If the p value was less than 0.05 the difference was considered significant.

Experiment 3. For our last experiment, we wanted to find the most descriptive features for each trait using a univariate f test.

7. Results

7.1. Experiment 1.

The accuracy and F1 results of all the models and feature combinations are shown in table 3. Summary of the model and features with the highest accuracy are shown in table 4. In the tables we use the following abbreviations: Agreeableness==A, Conscientiousness==C, Extraversion==E, Neuroticism==N, Openness=O.

Table 4: The F1 and accuracy scores for the best models and feature combinations.

Trait	Model and features	F1	Accuracy
E	XLNet + embeddings	0.613	0.604
N	XLNet + embeddings	0.468	0.556
A	XLNet + embeddings	0.683	0.542
C	SVM + Bag of words	0.570	0.589
O	XLNet + embeddings	0.488	0.636

As can be seen from the tables, all models perform similarly, but semantic models perform just slightly better on all traits.

7.2. Experiment 2.

Results of t tests can be seen in table 5. Only significant p values are reported. It is interesting to notice that Neurotic and Openness traits are significantly different from Extrovert and Agreeable traits, but are not significantly different from each other. This would suggest similar syntax styles between those two groups of personality traits, but more research is needed to confirm that.

Table 5: P value for t test on part-of-speech tags which are significant.

Traits	p value	PoS tag
N vs A	0.0119	verb
N vs A	0.0115	pronoun
E vs O	0.0141	noun
A vs O	0.0044	noun
E vs N	0.0020	punctuation
E vs O	0.0034	punctuation
A vs O	0.0154	punctuation

[2]https://scikit-learn.org/

[3]https://huggingface.co/transformers/

Table 3: The accuracy and F1 scores for all models and feature combinations (s: simple, d: detailed, nn: non-normalized, nbs: normalized by sum, nbsc: normalized by sentence count).

Model and features	E acc	N acc	A acc	C acc	O acc	E F1	N F1	A F1	C F1	O F1
Bag of words	0.536	0.539	0.507	**0.589**	0.594	0.592	0.455	0.598	0.570	**0.645**
PoS (s, nn)	0.539	0.513	0.523	0.515	0.567	0.625	0.462	0.641	0.558	0.575
PoS (d, nn)	0.538	0.529	0.515	0.507	0.554	0.614	0.474	0.639	0.518	0.518
NE (nn)	0.538	0.502	0.505	0.528	0.549	0.624	0.518	**0.644**	0.541	0.618
PoS + NE (s, nn)	0.555	0.495	0.491	0.531	**0.606**	0.642	0.461	0.619	0.544	0.644
PoS + NE (d, nn)	0.521	0.529	**0.523**	0.567	0.570	0.608	0.474	0.633	**0.588**	0.557
PoS (s, nbs)	0.546	0.497	0.529	0.539	0.567	0.593	0.498	0.643	**0.606**	0.611
PoS (d, nbs)	0.551	0.523	0.512	0.487	0.560	0.632	0.533	0.576	0.540	0.557
NE (nbs)	0.536	0.512	0.505	0.560	0.547	0.632	0.518	0.610	0.573	0.610
PoS + NE (s, nbs)	0.565	0.507	0.500	0.552	0.560	**0.633**	0.464	0.614	0.576	0.597
PoS + NE (d, nbs)	**0.570**	**0.534**	0.500	0.549	0.576	**0.635**	0.528	0.600	0.593	0.591
PoS (s, nbsc)	0.555	0.525	0.504	0.497	0.565	0.610	0.551	0.633	0.535	0.623
PoS (d, nbsc)	0.539	0.531	0.505	0.534	0.536	0.553	**0.553**	0.607	0.553	0.551
NE (nbsc)	0.557	0.502	0.504	0.549	0.554	0.618	0.532	0.616	0.551	0.625
PoS + NE (s, nbsc)	0.568	0.504	0.513	0.551	0.593	0.631	0.533	0.619	0.585	**0.649**
PoS + NE (d, nbsc)	0.541	0.536	0.510	0.560	0.581	0.573	**0.569**	0.600	0.577	0.616
XLNet	**0.604**	**0.556**	**0.542**	**0.586**	**0.636**	0.613	0.468	**0.683**	0.576	0.488

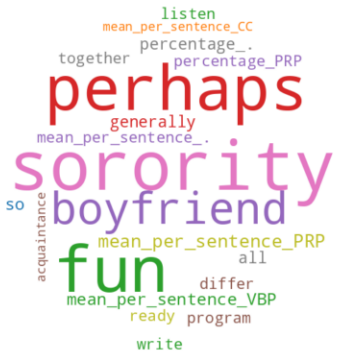


Figure 1: Most significant features for the Extraversion trait.

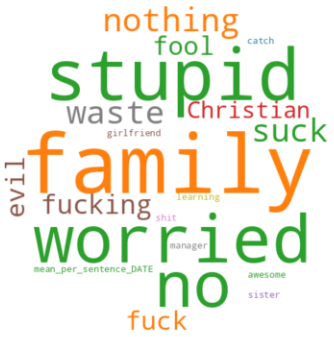


Figure 2: Most significant features for the Agreeableness trait.

7.3. Experiment 3.

Using word clouds we visualized the results of experiment 3. Other than being amusing, the figures are quite different from each other which shows a lot of potential for good classification models.

8. Conclusion

In this paper, we compared only syntax and only semantic models to see if one has an edge over the other. While there is certainly some information hiding in the syntax, semantics still seems to be the dominating factor. By looking at the most important features for each personality, we can clearly see the differences between them, and some are not that surprising. This promises a bright future for personality trait classification, and it seems that both syntax and semantics will be needed to create a powerful model.

References

John M Digman. 1990. Personality structure: Emergence of the five-factor model. *Annual review of psychology*, 41(1):417–440.

Lewis R Goldberg. 1993. The structure of phenotypic personality traits. *American psychologist*, 48(1):26.

Oliver P John, Eileen M Donahue, and Robert L Kentle. 1991. Big five inventory. *Journal of Personality and Social Psychology*.

Gregory Park, H. Schwartz, Johannes Eichstaedt, Margaret Kern, Michal Kosinski, David Stillwell, Lyle Ungar, and Martin Seligman. 2014. Automatic personality assess-

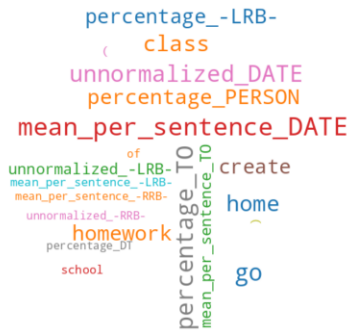


Figure 3: Most significant features for the Openness trait.

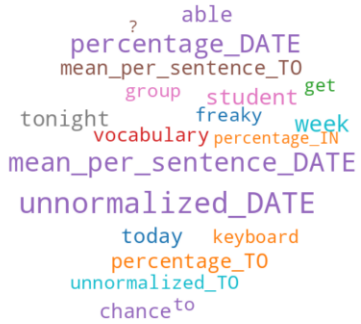


Figure 4: Most significant features for the Conscientiousness trait.

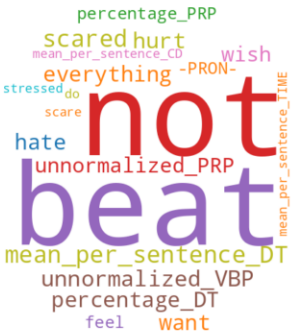


Figure 5: Most significant features for the Neurotic trait.

ment through social media language. *Journal of personality and social psychology*, 108, 11.

James Pennebaker and Laura King. 2000. Linguistic styles: Language use as an individual difference. *Journal of personality and social psychology*, 77:1296–312, 01.

Daniele Pizzolli and Carlo Strapparava. 2019. Personality traits recognition in literary texts. In *Proceedings of the Second Workshop on Storytelling*, pages 107–111, Florence, Italy, August. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Identification of Duplicate Questions

Nicolas Magne, Paul Lafont, Milan Pasquereau

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
paul.lafont@fer.hr, milan.pasquereau@fer.hr, nicolas.magne@fer.hr

Abstract

Many questions are posted daily on various forums such as Quora website, and the amount of questions asked can lead to doubles or multiples of sentences with same meaning. To reduce those copies, we propose three different methods to detect duplicate of sentences with same meaning, and to provide a solution to lower this amount.

1. Introduction

In this report, we will detail more and more precisely the work we made so far during this project, starting with an overview of the goal and the data of the project, to the solutions we provide.

The subject of this project is about duplicate questions on Quora website. Our goal was to implement a way to determine whether two questions had the same intent or not.

To reach our goal, we had to determine how we would collect a sufficient amount of Quora questions to study it. The subject is based on a Kaggle competition, for which a dataset was given with the following format: question1, question2, 0 or 1 (duplicates or not) We used then over 2 millions of pairs of questions, in which some had the same intent, and some were completely different, and we could separate this huge dataset into many others to have different testings sets.

2. Pre-processing

2.1. Analysis and Data Filtering

While it is necessary to use some kind of word embeddings dictionaries for this project, this requires some previous steps to normalize the data.

2.1.1. Lowercasing

Before any process, we needed that every word or sentence would be treated the same way regardless of importance a word can have written in Capital letters, or to create some troubles between two same words written differently. We wanted to consider for example: "tree", "TREE", or "TrEe" as the same word to make it easier to analyze.

2.1.2. Removing Stop Words

As in many Natural Language Processing, we might consider to filter all the words useless for detecting the intent of the full questions. For all our models, we chose to remove those stop words.

2.1.3. Part-of-Speech tagging

Using the feature extraction approach we will explain later, we thought it would be unprecise to compare full sentences. To improve the accuracy of the model we tried to implement a part-of-speech tagging, which will compare only words from same category. Comparing nouns with nouns, verbs with verbs would help to detect more efficiently if two questions have the same meaning.

2.1.4. Lemmatization

Considering the amount of pairs of questions our dataset provides, the execution time of the models were really long, comparing many differents words, and sometimes many versions of a word, as "love", "lovely", "lover"... This words could be ambiguous for the models and mislead to different meaning. In all our models, to save this amount of time and to be sure to don't miss a matching pair, we added a lemmatization process which reduce each words to its shortest/simplest version, and allow us to find pairs of questions with same words or same subjects.

2.1.5. Padding

In our latest model, using the neural network approach, our data needed to fit the model with a "standard" size. All questions and words from the Quora questions were messy and had different length. To solve this problem, we defined a standard size and filled shorter words/sentences with 0 to match the input needs.

3. Words Embeddings

For this project word embedding is kind of a must-have. Word embedding can be used to determine whether two words meanings are somewhat related to one another. We then compared different approach of embedding and which one would suit better to our solution.

- **Word2Vec:** Word2Vec is using full texts during its training phase.
- **FastText** FastText is using N-grams during its training phase
- **GloVe** GloVe is using the number of co-occurences two words have in a corpus. We chose a model pretrained on wikipedia, because it insures us that the model is considering two words as similar when they have the same meaning. For example the name "Atelopus zeteki" (kind of frog), will have a close similarity with the word "frog" because it is more cited in its wikipedia article.

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 52–55, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

4. Feature extraction approach

4.1. Feature Extraction - Pairwise Distance

Our first method to solve this problem, with the easiest tool we had was to check the accuracy and the similarities of two questions using simple pairwise distance.

We computed several pairs of questions using either Jacquard Similarities or Cosine Similarities, but as we expected the results are not that satisfying. Some questions can shows evident results for pair which have only one or two words differences, but results are more messy when the order of words changes or synonyms are used.

From this approach we conclude that the pairwise similarities are sufficient for related questions, but are not efficient at all because the meaning of the sentences isn't taken into consideration.

4.2. Modeling - Logistic Regression

After the bad results presented by using pairwise distances, we wanted to implement a linear model for classification: the logistic regression.

In this model, we used two python's library used for NLP: NLTK[1](**?**) and Scikit-Learn[2](7), the first used for all the pre-processing steps, and the second for the model and the training and testing phases.

We first removed all the stop words contained in the NLTK corpus of stop words. Then we Switch all our pair of questions into lists on Token to manipulate them easier.

All the tokens passed a POS tagging process and were compared with tags as Nouns, Adjectives, Adverbs or Verbs.

On top of this process, we define a computation to determine the overlap of words of same category between lists of words from two different questions.
At this step our data were fitted to matched the model requirement, and we trained the model with half the questions pair the dataset contains. We used around 1 million pair of questions as Training sets, and the other million as a testing set, to verify the correctness of the trained model.

After playing with the hyper-parameters and applied Logistic regression to the values obtained by training the model and the raw values of our datasets, we finally achieved to reach 66.3% of accuracy.

5. Neural network approach

In this section we will see how we implemented an LSTM neural network and optimized its hyperparameters .

[1]https://www.nltk.org/

[2]https://scikit-learn.org/stable/

5.1. LSTM-Network model

LSTM-Networks are a popular choice for natural language processing and show a strong performance in many tasks. The neural network architecture was designed such that it follows deep neural networks architecture rules while preventing some drawbacks of such designs.

We designed 2 models:

- A simple model to optimize the sophisticated model hyperparameters (model A)
- A sophisticated model to find the best perfomances with 2 LSTM-Networks (model B)

We did so because it was impossible on our computers to find the right hyperparameters for model B.

5.1.1. Model A: LSTM_GLoVe_HPT_BO

The model given in Figure 1 is using GloVe embeddings and is designed for hyperparameter tuning (HPT) based on Bayesian optimization (BO).

This must follow some rules to be optimized resource-wise, other architecures are possible but we absolutly needed cuDNN kernel optimizations because otherwise the training time would be too long. cuDNN kernel optimizations require some choices regarding LSTM layers.

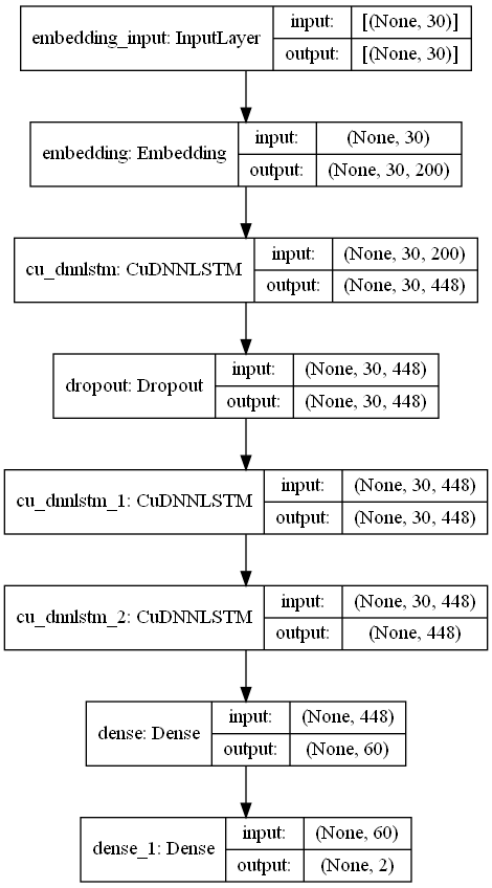


Figure 1: LSTM_GLoVe_HPT_BO model.

This model is made of 7 layers:

1. Embedding layer: used as the input layer to load GloVe embeddings

2. CuDNNLSTM layer: First LSTM layer(7)

3. Dropout layer:
   "Every LSTM layer should be accompanied by a Dropout layer. This layer will help to prevent overfitting by ignoring randomly selected neurons during training, and hence reduces the sensitivity to the specific weights of individual neurons. 20% is often used as a good compromise between retaining model accuracy and preventing overfitting."

4. CuDNNLSTM layer: Deep neural networks (DNN) are used to solve more complex problems, the difference with normal neural networks (NNN) lies in the number of hidden layers (2 or more for DNN and 1 for NNN). More layers can be better but also harder to train.

5. CuDNNLSTM layer: DNN

6. Dense layer: dimension reduction

7. Dense layer: dimension reduction

We will see the layers hyperparameters in the next sections

5.1.2. Model B: LSTM_GloVe_QS

The model given in Figure 2 is using GloVe embeddings and is designed for performances. It differs from model A by implementing question separation (QS). The idea is to merge the models for question1 and question2.

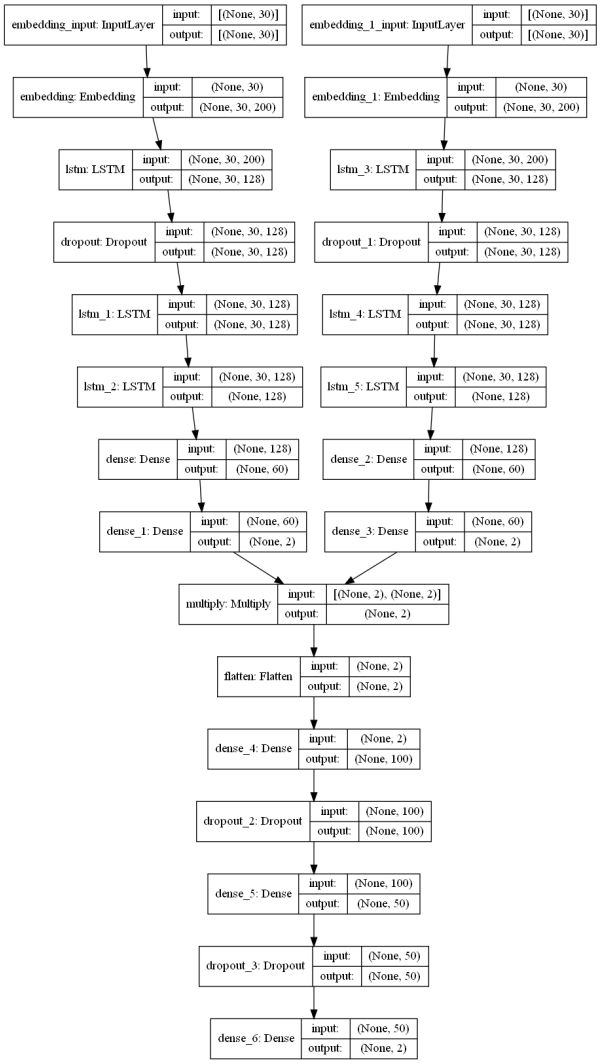


Figure 2: LSTM_GLoVe_QS model.

This model is made of 2 merged model A, after which a repetition of dense and dropout layers. The number of repetitions is optimized by grid search. Also the learning rate is optimized with the Bayesian optimization . If the learning rate is set too low, the model will miss important patterns in the data, but if it is too large, the model will find patterns in accidental coincidences too easily.

5.2. Hyperparameters optimization

While creating a neural network the hyperparameters need to be set. There are no general rules on fixing these values and it is mostly done through trial and error methods. Multiple methods exists with different complexity, there are basic techniques(7):

- Grid search: Test all possible combinations of hyperparameters and take the best
- Random search: Pick some random combinations and take the best

We wanted to try a more sophisticated method: the Bayesian optimization after reading Google's article on how they optimize their own networks

We chose which techniques to use based on the fact that we wanted good but not perfect hyperparameters combinations so that the time to train the models wouldn't take weeks.
This is what we came up with:

- Grid search: for the number of Dense and Dropout layers of the merged model. Between 1 and 4.
- Bayesian optimization: for the number of LSTM cells per layer and the model learning rate.

5.2.1. Hyperparameters optimization results

After testing multiple combinations, the best hyperparameters found are:

- Learning rate: 0.001
- Cells per layer: 128
- Dense / Dropout layer repetition: 2

We didn't describe the process but we found that training the model for 7 epochs gave the best results

5.3. Results

After testing multiple options, the best model achieved was found with the following options: no early stopping of the model training, not removing stop words, 7 epochs.

Model	Accuracy
LSTM_GLoVe_HPT_BO	63%
LSTM_GloVe_QS	75%

Table 1: Models comparation.

6. Conclusion

After this projects, we definitely figured out the problem of duplicate sentences was harder than expected.

The simplest model we tried were not efficient enough to determine a decent rate of duplicates, and even with solid models the maximum accuracy we obtained is up to 75%.

The performance of a stronger model than our will fully rely on the tuning of the hyperparameter of the model.

7. Related Work

There is a plethora of related work on pairs questions. In particular, an approach using BERT has been used by many developers. This approach has a fairly good accuracy of 67.14%.

BERT (Bidirectional Encoder Representations from Transformers) provides dense vector representations for natural language by using a deep, pre-trained neural network with the Transformer architecture.

Also, an approach using gloVe and LSTM (Long short-term memory) had good results with an accuracy of 67.45%. This approach was developed by Debasis Samal, an Indian engineer.

References

Nils Reimers and Iryna Gurevych. *Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks* German Institute for Educational Research, 2017

Bird, Steven, Edward Loper and Ewan Klein *Natural Language Processing with Python.* O'Reilly Media Inc, 2009

Puneith Kauln, Daniel Golovin and Greg Kochanski. *Hyperparameter tuning in Cloud Machine Learning Engine using Bayesian Optimization*. Google, 2017

Pedregosa et al. *Machine Learning in Python* JMLR 12, pp. 2825-2830, 2011

Tensorflow CuDNNLSTM layer requirements. *tensorflow.org/api*$_d ocs/python/tf/keras/layers/LSTM$

Prabowo and Herwanto. *Duplicate Question Detection in Question Answer Website using Convolutional Neural Network.*

McCreery et al *Effective Transfer Learning for Identifying Similar Questions: Matching User Questions to COVID-19 FAQs.*

Stressformers: Transfering Knowledge for Stress Analysis in Social Media

Lovro Matošević, Filip Sosa, Marko Gašparac

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{lovromatosev,filip.sosa,marko.gasparac50}@gmail.com

Abstract

Stress is a natural physical and mental reaction to life experiences which everyone experiences from time to time. Even simple things like everyday responsibilities can trigger stress. While it can be a motivator, too much of it can cause serious negative health issues which is why it's crucial to detect symptoms of stress as early as possible. In this paper, we explore the efficiency of a transfer learning approach for stress classification using Dreaddit, a text corpus consisting of lengthy multi-domain posts from Reddit. We fine tune a number of pre-trained transformer models and compare their efficiency on the problem of stress classification.

1. Introduction

Following the tremendous advance of information and communication technology, for the past two decades we witness the immense growth of Internet and social media. Social networks have changed the way we communicate and interact with each other, and have had a large impact on our social society. They are nowadays in our constant presence, and we use them not only to exchange textual messages with our friends, colleagues and acquaintances, but to share our everyday happenings and experiences with others through photos, videos and other multimedia.

The feeling of importance of our social status, worry on how we are performing compared to the others and whether or not we are leaving a good first impression may further contribute to the development of stress - a reaction to extant and future demands and pressures. Even though stress is a highly subjective experience whose definition, effects and most importantly situations that cause it vary greatly from person to person, an existence of widespread experiences of chronic stress that impacts physical and mental health has already been demonstrated. This brings an importance of studying different methods of early stress detection.

The enormous amount of textual content that is generated each day on social media serves a great potential for conducting scientific research on detecting textual patterns that indicate that the author might be under the influence of stress. In this work, we use Hugging Face transformers (Wolf et al., 2020), a state of the art Natural Language Processing tool that provides thousands of pretrained models to perform tasks on texts such as classification, information extraction, question answering, summarization, translation and text generation. We applied several such models on Dreaddit - a dataset of lengthy social media posts. They were posted in five different categories and include both stressful and non-stressful text. Each post expresses stress in a different way. The dataset was collected on Reddit, a network of communities based on people's interests where users can write posts, their comments, and like or dislike different content.

Our main point of interest is demonstrating how these state of the art models perform assessing stress. Thus, we call them Stressformers. We discard any metadata and hand crafted features from the dataset and focus solely on a snippet of text, its label and knowledge that has been transferred from the previous tasks these models were pretrained on. We compare the performance of transformers of different architectures and observe how do these truly end to end systems compete with traditional shallow approaches. We also provide an insight on some of the examples these methods did not perform well on.

In the remainder of this paper, we review the related work, describe the dataset we have used, the architecture of different models we have experimented with and the experimental setup itself. Finally, we present the results and conclude with plans for further research.

2. Related Work

Most of our work was influenced and inspired by Turcan and McKeown (2019). As a result of their research, Dreaddit was made publicly available. They encourage the use of their dataset for detecting the presence of stress and hope it will facilitate the development of models for other various applications such as diagnosing physical and mental illness, gauging public mood and worries in politics and economics, and tracking the effects of disasters.

We have used Dreaddit to train the Stressformers, and we have compared the obtained results with both the discrete and neural supervised models that had emerged as a result of their work. Finally, they analyze the content of their dataset and explain how it reflects the performance of their models. This is something we strive to do for the Stressformers as well as it provides great insight into the problem of stress detection.

3. Dataset

Dreaddit consists of 3,553 posts that were scraped from Reddit between January 1st, 2017 and November 19th, 2018. They were collected from subreddits where members were likely to discuss about topics that usually cause stress such as interpersonal conflicts, mental illnesses or financial needs. The majority of posts originate from r/anxiety, r/assistance and r/ptsd and fall under one of the following domains: abuse, anxiety, financial, PTSD or social. The resulting dataset is fairly balanced, with 52.3% of the data (1,857 instances) labeled stressful.

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 56–59, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

The average length of a post is 420 tokens. The lengthy nature of these posts allow more detailed study of the causes and effects of stress. Each text example is assigned a binary stress label after being annotated by English speaking Mechanical Turk Workers. Dataset also contains a set of various other features that emerged as a result of data analysis. We discarded those as we had not made use of them.

4. Models

In this section we will give a brief overview of the different transformer models we tried out on our task. All of the models we used are from Hugging Face Transformers repository which provides general-purpose architectures (e.g. BERT).

4.1. BERT

The BERT model was proposed by Devlin et al. (2019). It's a bidirectional transformer pretrained on a large corpus consisting of Wikipedia and the Toronto Book Corpus by using a combination of masked language modeling objective and next sentence prediction.

4.2. ConvBERT

The CONVBert model tries to tackle BERT's problem of heavily relying on the global self-attention block (Jiang et al., 2021). The model itself contains a span-based dynamic convolution to replace the mentioned self-attention heads to directly model local dependencies.

4.3. RoBERTa

The RoBERTa model builds on BERT and modifies its key hyperparameters (Liu et al., 2019). It removes the next-sentence pretraining objective and trains with much larger mini-batches and learning rates.

4.4. I-BERT

The I-BERT model is quantized version of RoBERTa running inference up to four times faster (Kim et al., 2021). It is based on lightweight integer-only approximation methods for nonlinear operations. It performs an end-to-end integer-only BERT inference without any floating point calculation.

4.5. XLNet

The XLNet is an extension of the Transformer-XL pretrained using an autoregressive method in order to learn bidirectional contexts by maximizing the expected likelihood over all permutations of the input sequence factorization order (Yang et al., 2019).

5. Experimental Setup

Our goal was to examine and compare 5 different models which are described in section 4. The dataset used for training consisted of 2,838 data points, of which 51.6% were labeled stressful. Furthermore, we break off random 10% of the training set for validation.

Table 1: **Results.** Precision, recall, and F1-score for our supervised models. The best model in our case is RoBERTa with an F1 score of 82.68 on the test set.

Model	Precision	Recall	F1
BERT	77.65	75.33	76.47
ConvBERT	**79.45**	71.27	75.14
RoBERTa	76.49	**89.97**	**82.68**
I-BERT	74.82	88.61	81.14
XLNet	77.51	87.80	82.33

5.1. Preprocessing

The main tool for preprocessing data while using pretrained transformer models is a tokenizer. Each of the models that we are using comes with an associated tokenizer. The tokenizer is used for splitting the given text into words, or parts of words. After that it converts those tokens into numbers in order to build tensors out of them which will be used by our models. The tokenizer also adds special tokens to the provided input, which are different for some of the models that we are using.

In addition to using a tokenizer, we also considered using a spelling correction tool because we noticed there are a number of words which are spelled incorrectly. After experimenting with some of the free tools such as Symspell[1] and Spacy's Contextual Spell Check[2], we concluded that we wouldn't be using spell correction. The main reason for that is inadequate results. Many of the users' spelling errors proved to be too difficult for the spelling correctors.

5.2. Model Configurations

We decided to use the default values for most of the parameters. One important hyperparameter is the maximum length of the tokenized text. This varies from model to model because of the different tokenizers that they use. We decided to set this value to the minimal possible value for each model according to the longest post from our dataset due to the time efficiency.

As for the optimizer, we decided on using AdamW (Loshchilov and Hutter, 2019). After experimenting with different learning rates we found that the best performance is with a learning rate of 10^{-5}. We also experimented with freezing some of the models layers but we concluded that the results are better without freezing any of the layers.

6. Results and Discussion

The results of our experiment are shown in table 1. The best model in our case is RoBERTa with an F1 score of 82.68 on the test set. The best recall score of 89.97 also belongs to RoBERTa. We find that recall in this case is much more important than precision. We also note that both BERT and ConvBERT achieve comparably worse results than the other three models. It's also interesting to note that both of them have significantly lower recall scores.

[1] https://github.com/wolfgarbe/SymSpell

[2] https://spacy.io/universe/project/contextualSpellCheck

Table 2: **Error Analysis Examples.** Examples of test samples all models failed to classify correctly.

Text	Label	Agreement	Subreddit Name
I have a question about my ex who has a past of violence against women. I was never warned about it but I found out he was violent and I left. His ex has a full life restraining order against him.Now he is on probation for assaulting a police officer for 3 years in the past year he has gone to jail three times for domestic violence. His latest trip to jail was last week for domestic violence his third time. I was wondering what do you think his punishment will be since he's not learning his lesson from the punishments given to him and he just doesn't care.	Not Stress	100%	domesticviolence
She's the first person I've ever really opened up to. I haven't told her everything about whats happened, but she does know about my anxiety (which I get from my PTSD) and she reacts sportively to it. To some extent, I let me be "myself" around her, whatever I am. She's moving. She's moving to Maryland.	Stress	80%	ptsd
I'm asking yall how can I live life properly? Immediately after I threw up in year 2, I never feared it happening again. I admit when this happened I was in tears, and same in year 3- I was crying next to my mum by the toilet but it was over quick and again- I was eating chocolate again the day after. But now, nausea? Stop eating for the day.	Not Stress	60%	anxiety

Table 3: **Intersection over union.** We compute intersection over union between sets of examples in which each model failed in order to see similarity between them.

	BERT	ConvBERT	RoBERTa	I-BERT	XLNet
BERT	1	0.5265	0.3716	0.3571	0.4027
ConvBERT	0.5265	1	0.3376	0.304	0.3788
RoBERTa	0.3716	0.3376	1	0.5561	0.5530
I-BERT	0.3571	0.304	0.5561	1	0.5235
XLNet	0.4027	0.3788	0.5530	0.5235	1

Table 4: **Average Agreement.** We compute average agreement of failed examples for every model.

Model	Avg. Agreement
BERT	63.74
ConvBERT	63.24
RoBERTa	58.71
I-BERT	58.47
XLNet	56.40

Table 3 shows intersections over unions between sets of examples in which our models failed. We note that in spite of high accuracies of our models, most of them fail on similar examples. For example, 52.65% of examples on which BERT and ConvBERT fail are the same.

We performed an error analysis on all of the models. Generally, the examples where our models failed are mostly ones with low annotator agreement. That can be seen in table 4. There is also some correlation between a model's score and the average annotator agreement of examples where the model failed. We note that models with a lower F1-score (e.g. BERT and ConvBERT) have a higher average agreement than models with a higher F1-score (e.g. I-BERT and RoBERTa).

There are 49 data points, or 6.85% of the test dataset where all 5 models failed. A couple of these problematic examples can be seen in table 2. Many examples that are false positive contain a number of stress related words. Also, often these false positive examples are actually stories where another person aside from the post's author is the focus. Furthermore, most of the false negative examples contain implicit expressions of stress, for example the authors talk about stressful experiences in a retrospective, positive way. We expected this prior to evaluating our experiment. This is because the transformer models we used primarily use textual, lexical features for classifying stress.

As for the comparison with the work of (Turcan and McKeown, 2019), we note that all of our models achieved comparable results to theirs. Their best performing discrete model, logistic regression with Word2Vec embeddings achieves an F1-Score of 79.80 with a recall score of 83.2. In addition to that, they also tried using a pretrained BERT model which achieved an F1-score of 80.65 and a

recall score of 86.99.

7. Conclusion

Stress classification is a complex problem. Posts where the author's story isn't about himself and posts where the author talks about a stressful experience in a positive way are just an example of the problem's complexity.

In this paper, we compared a number of transformer models on the task of stress analysis in Reddit posts. We set the current baseline at 82.68% F-Score. We conclude that stress classification is mainly a lexical problem and that it greatly benefits from using a transfer learning approach.

Improvements can be made by further experimenting with models' hyperparameters. It would also be interesting to research whether there could be any improvements by combining one of the transformer models with some of the other, non-lexical features.

References

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.

Zihang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. 2021. Convbert: Improving bert with span-based dynamic convolution.

Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. I-bert: Integer-only bert quantization.

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.

Ilya Loshchilov and F. Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Elsbeth Turcan and Kathy McKeown. 2019. Dreaddit: A Reddit dataset for stress analysis in social media. In *Proceedings of the Tenth International Workshop on Health Text Mining and Information Analysis (LOUHI 2019)*, pages 97–107, Hong Kong, November. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Going to the Depth for Celebrity Prediction

Patrik Matošević, Ivan Križanić, Mario Zec

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{patrik.matosevic,ivan.krizanic,mario.zec}@fer.hr

Abstract

Extracting information about users has become one of the greatest interest of many parties, mainly because marketing runs the Internet world. To improve such tasks, a branch of research called user profiling appeared in natural language processing. A text carries a lot of information, so it is only natural to try to exploit information stored in a text to predict some characteristics about people who wrote it. This paper tries to replicate results from PAN-2019 and to check if some conclusions from that competition still stand today. Our focus is mainly to compare how well two different approaches perform on the same task, and also to explore if known techniques for determining feature importance have real value in the transparency of deep learning models. Additionally, we will try to predict which examples will deep learning fail to predict and use that information to determine which features are problematic for deep learning.

1. Introduction

Textual content played a major role in humankind since the invention of the written word. Since then, many things have changed, but the text remains one of the most important sources of information and various types of content. With the rise of social networks, a whole new category of textual content appeared. Platforms such as Twitter and Facebook allow users to express their feelings, political views or to share with the world their thoughts on every subject imaginable. That makes posts from social networks extremely valuable and interesting data to study. It allows research of various psychological and sociological topics, for example, psychological disorders detection or bullying detection. It can also serve as a source for studying meaning, sense or sentiment in text.

The use case we are most interested in is user profiling based on posts, concretely on tweets. We are currently living in a time where ads and recommendation systems have a major role in almost every business on the Internet and, as always, money and profit attract the interest of a wide spectre of people, including researchers. It is not surprising that reliable and comprehensive techniques for the task of user profiling are in high demand. Nowadays, a great number of *state-of-the-art* solutions for NLP (natural language processing) problems are based on deep learning techniques, especially if there is enough data to successfully implement such models. It comes as a bit of surprise that many participants of the PAN-2019 contest reported under-performing deep learning models compared to classic machine learning techniques (Wiegmann et al., 2019). That fact motivated us to test if traditional approaches still outperform deep learning in user profiling task, and if so, how big is the difference in performance and what are some other pros and cons of using deep learning versus traditional techniques. We also reflected on the problem of interpreting deep learning models, because the fact that deep learning models often work as a black box is considered to be a drawback. We tried to implement traditional machine learning models based on best-performing implementation from PAN-2019 (Radivchev et al., 2019), and for the deep learning approach, we used BERT pre-trained on Twitter data (Nguyen et al., 2020).

2. Related work

Classification of various user attributes based on text attracts researchers for more than a decade. The main focus is often directed towards age, gender and personality. At first, long texts with higher quality contents were used, but recently shorter and simpler texts such as comments and posts from social networks, mainly Twitter and Facebook, became the main resource for research. Author profiling is present in various competitions, and it has been present in the PAN competitions since 2013, where only gender and age were considered (Rangel et al., 2013). Since then, mainly machine learning approaches were used, often classic models such as logistic regression and SVMs (Rangel et al., 2015), but in the more recent years deep learning received great interest (Rangel et al., 2018). In the edition of the PAN competition we are focusing on, only one submission was based on deep learning, while some others mention failing to achieve comparable performance with deep learning compared to classic methods. The only successful attempt with deep learning in PAN 2019 was based on transfer learning (Pelzer, 2019) which was motivated by previous work on Twitter classification (Yang et al., 2017).

As for a question of deep learning models interpretability, there are attempts to use gradients of class scores with respect to inputs to find the most valuable input aspects for classification (Simonyan et al., 2013). Although mentioned paper uses this method for the image classification task, the same idea can be applied to natural language processing. Attention is often mentioned to provide a level of transparency to models, in a way that words with high attention have to be responsible for prediction, but it was shown that attention should not be used as an explanation (Jain and Wallace, 2019). Because of the reported lower performance of deep learning models, we would like to try if it is possible to determine which features are causing deep learning models to fail when traditional approaches do well.

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 60–64, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

3. Experiment setup

3.1. Dataset

Dataset from PAN-2019 competition is used in this paper. Dataset was sampled from the Webis Celebrity Profiling Corpus 2019, which originally contains 71,706 celebrities. The final dataset has a smaller number of celebrities, but still quite a large number of 48,335 with each celebrity having 2,181 tweets on average (Wiegmann et al., 2019). Dataset labels contain four different features: gender, fame, occupation and age. *Gender* label contains values *male, female* and *non-binary*. *Fame* label contains three different levels of fame, based on number of Twitter followers, those three being: *rising* for less than 1,000 followers, *star* for less than 100,000 followers, and finally *superstar* for more than 100,000 followers. *Occupation* label was made by grouping 1,379 occupations in eight classes: *sports, performer, creator, politics, manager, science, professional, religious*. *Age* label was not divided into age groups and year from the day of birth was used instead.

As for the number of each possible labels, some are disproportionally represented in the dataset. The distribution of labels can be seen in figure 1. Some extremely disproportional labels are for example *non-binary* in *gender* label and *science* in *occupation* label. Train and test split retain proportions of labels and the average number of tweets per user, making them well-balanced splits.

In the training of our models, a subset of the original dataset was used, both because of resource limitations and consistency with previous work we were trying to replicate.

3.2. Metrics

Performance measure used in PAN-2019 is $cRank$ (1), which is implemented as harmonic mean of macro-averaged $F_{1,T}$ scores for $T \in \{gender, fame, ocupation\}$ (2). *Birthyear* had different metric because labels were not divided into classes, instead, score m was used to assess if the label is correctly predicted in a way that if a prediction is inside of m-window of the true year then the prediction is considered as correct. The harmonic mean was used to tackle the problem of the imbalanced distribution of classes (3).

$$cRank = \frac{4}{\frac{1}{F_{1,fame}} + \frac{1}{F_{1,occupation}} + \frac{1}{F_{1,gender}} + \frac{1}{F_{1,birthyear}}} \quad (1)$$

$$F_{1,T} = \frac{2}{|T|} \cdot \sum_{t_i \in T} \frac{precision_{t_i} \cdot recall_{t_i}}{precision_{t_i} + recall_{t_i}} \quad (2)$$

$$m = (-0.1 \cdot truth + 202.8) \quad (3)$$

3.3. Preprocessing

Preprocessing was made in a classic manner for tweets analysis, and it was implemented to produce the same result as the best submission on PAN-2019 (Radivchev et al., 2019). It consists of several steps:

- removing all special symbols except @ and #
- replacing all hyperlinks with <url>

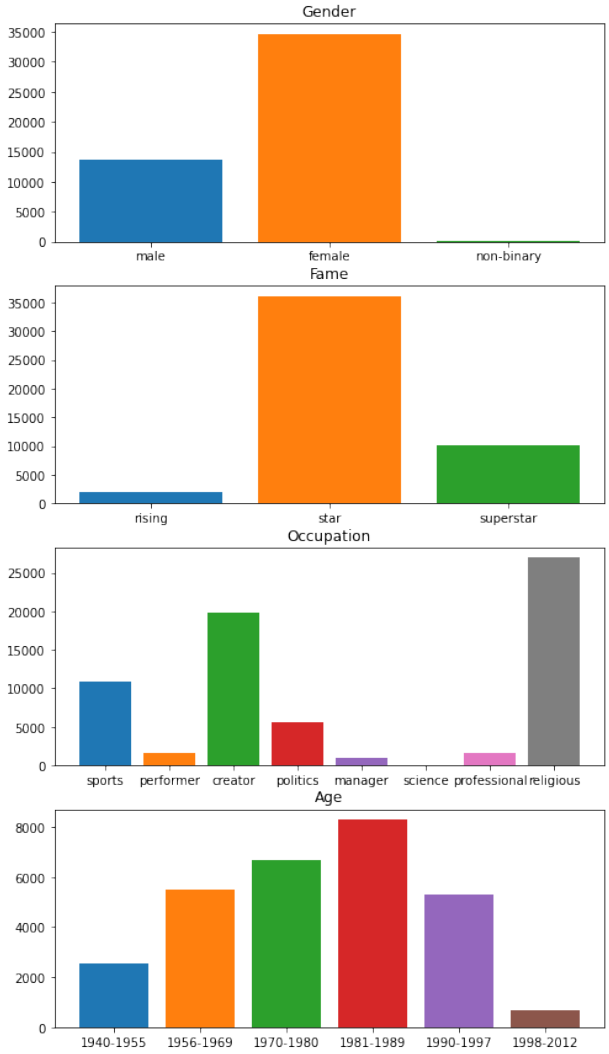


Figure 1: Labels distribution in whole dataset.

- replacing all user tags with <user>
- lowering all characters

Finally, all tweets were tokenized using NLTK Tweet Tokenizer (Loper and Bird, 2002).

4. Models

Two approaches were used in implementing a solution for the user profiling task. The first approach was based on traditional machine learning techniques, mainly on support vector machine (SVM) (Cortes and Vapnik, 1995) and logistic regression. The second approach was based on deep learning, precisely BERT pre-trained on Twitter data (Nguyen et al., 2020).

We have not used the whole dataset for training the models, instead, only a hundred tweets were randomly sampled for each user. Previous work has shown that the number of tweets greater than 500 does not contribute to increased performance on test data, but that number was too large to conduct experiments with available resources.

Table 1: BERT hyper-parameters

Hyper-parameter	Value
Batch-size	32
Learning rate	$3 \cdot 10^{-5}$
Optimizer	AdamW
Scheduler	linear with warmup
Warmup	500 steps
Linear LR decay	from step 500 till the end
Epochs	2

4.1. Traditional approach

In the view of previous work, the traditional approach seemed to be a way to go when implementing a user profiling model. We used the most successful submission from PAN-2019 as our reference point for implementing SVMs and logistic regression models. As already mentioned in the section3.3., we used the same preprocessing pipeline. For each class, a different model was used. For *gender* label we used multinomial logistic regression with newton-cg solver, and for other labels, we used SVM with linear kernel. We used the same class weights as mentioned submission, to deal with the problem of unbalanced labels.

4.2. Deep learning approach

For the deep learning approach, we used both BERT general pre-trained model together with an additional model pre-trained specifically on tweets. This seems to be a novelty as we did not found any mentions of using BERT in previous related work. The reason we have chosen BERT as our representative for deep learning models lays in the fact that a great number of current *state-of-the-art* implementations for a wide spectrum of problems use BERT (Ruder, 2021). As far as fine-tuning is concerned, hyper-parameters were chosen in the spirit of standard practices and can be seen in table 1. AdamW was chosen as optimizer because the weight decay and learning rate can be optimized separately (Loshchilov and Hutter, 2017). It is also worth mentioning that all tweets were truncated to a maximum size of 64 tokens.

4.3. Gradients for feature importance

Besides trying to implement models and comparing the performance of the traditional approach and deep learning on the problem of user profiling, we also wanted to test how well could we mimic the interpretability of the traditional approach on our deep learning model. As we mentioned in the section 1., there are attempts to use gradients of class scores with respect to inputs in an attempt to find the most valuable features used in prediction (Simonyan et al., 2013). We implemented a logistic regression model that uses unigrams as features to predict only the *gender* label. We then used weights of logistic regression to determine which unigrams are of the greatest significance for prediction, with assumptions that those are the ones with the highest weight assigned to them.

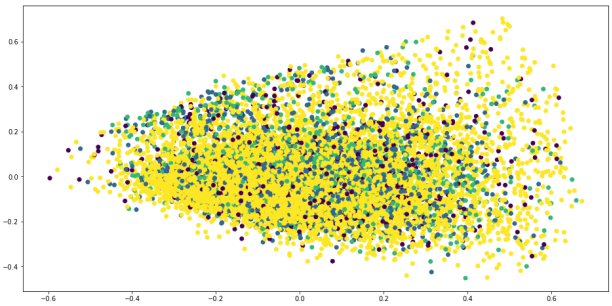


Figure 2: PCA of prediction success based on unigram features

4.4. Linear regression for predicting problematic features

We implemented linear regression that uses unigram features as input and tries to predict if the deep learning model will fail to predict the correct label. The learning is done on the specifically crafted target labels. The target is set to one if the deep model gives an incorrect prediction while the shallow model gives the correct prediction, and it is zero otherwise. We assume that if learning would be successful, then we will be able to use learned weights to determine which features are problematic for the deep learning model.

Performance of model which is used as a reference point for calculating differences in prediction is displayed in table 2. The features in this model were TF-IDF (term-frequency-inverse-document-frequency) vectors from 5000 most common unigrams from concatenated tweets (preprocessed with already mentioned preprocessing pipeline). We used the Lasso model (linear regression with L1 regularization) to perform an implicit feature selection through the model.

PCA plot of features in linear regression coloured by model correctness (both models give a correct prediction, both fail or one fail and the other one gives correct output) is shown in figure 2.

5. Results

The performance of our models is below reported performances on PAN-2019, but it is in line with expectations as we trained on a smaller number of examples. Nevertheless, we kept a consistent number of examples throughout tests to make as fair a comparison as possible. In the accordance with negative experiments reports when using deep learning, our deep learning BERT model fails in comparison with our traditional implementation. As it can be seen in table 3, BERT falls behind traditional models (SVMs and logistic regression) by great margin, both in final *cRank* metric, and in individual $F_{1,macro}$ metrics. We also experimented with the base BERT model and, as expected, it performed worse than one that is pre-trained on Twitter data. The worst result was achieved with the base BERT model when preprocessed data was used, which showed that BERT's internal preprocessing already does all the work necessary for best performance.

Linear regression gave us some useful insights about features that cause the deep learning model to fail. We con-

Table 2: Performance of SVM trained on unigram features

Data	cRank	$F_{1,gender}$	$F_{1,fame}$	$F_{1,occupation}$	$F_{1,birthyear}$
test1	0.44	0.55	0.50	0.39	0.36
test2	0.44	0.56	0.47	0.39	0.39

Table 3: Results on test2 dataset

Approach	cRank	$F_{1,gender}$	$F_{1,fame}$	$F_{1,occupation}$	$F_{1,birthyear}$
Traditional (LR & SVM)	0.46	0.56	0.47	0.48	0.38
BERT (bert-base-uncased)	0.32	0.34	0.38	0.35	0.25
BERT (*Bertweet*)	0.37	0.45	0.32	0.41	0.34

cluded that when applying L_1-regularization during training of linear regression model, we are left only with meaningful features, so we conclude that those features must be responsible for failure. With those insights, we implemented a pipeline that gives us the wanted number of what are considered to be the most significant tokens that caused the wrong prediction. Linear regression was used to conduct statistical testing, but because of significant weights magnitude, we have not conducted any statistical tests and based our conclusions mainly on intuition.

Implementation of feature importance extraction showed us that the deep learning model sometimes overfits on unusual tokens such as <url>, <user>, <rt>, especially when input tweets are shorter. We also determined which bigrams are considered to be important in traditional models based on weights. Some examples for those bigrams for *female* class in *gender* label are: (on, her), (women, in), (the, womens), (xx, sep), (for, her) and for *male* class: (bro, sep), (man, sep), (my, wife), (mate, sep), (my, boy). We can see that those unigrams are quite intuitive.

6. Conclusion

Our experiments showed that deep learning models indeed produce less accurate models for user profiling compared to traditional approaches such as SVM or logistic regression, and the observed difference was visible in all comparisons, both in final *cRank* metric and in individual $F_{1,macro}$ metrics, even when we were using model specifically pretrained for usage on Twitter data.

It seems that linear regression could give us some insights into the importance of features that cause failure in deep learning models where traditional models gave correct prediction. This observation should be tested and if shown as correct, there could be some useful implications in boosting deep learning models performance in some situations.

With our feature importance analysis we concluded that feature importance based on weights seems to be intuitive in traditional models, while in deep learning, importance based on gradient analysis showed us that unusual tokens often end up being the most important feature in prediction.

References

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Sarthak Jain and Byron C Wallace. 2019. Attention is not explanation. *arXiv preprint arXiv:1902.10186*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *CoRR*, cs.CL/0205028.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online, October. Association for Computational Linguistics.

Björn Pelzer. 2019. Celebrity profiling with transfer learning. In *CLEF (Working Notes)*.

Victor Radivchev, Alex Nikolov, Alexandrina Lambova, L Cappellato, N Ferro, DE Losada, and H Müller. 2019. Celebrity profiling using tf-idf, logistic regression, and svm. In *CLEF (Working Notes)*.

Francisco Rangel, Paolo Rosso, Moshe Koppel, Efstathios Stamatatos, and Giacomo Inches. 2013. Overview of the author profiling task at pan 2013. In *CLEF Conference on Multilingual and Multimodal Information Access Evaluation*, pages 352–365. CELCT.

Francisco Rangel, Paolo Rosso, Martin Potthast, Benno Stein, and Walter Daelemans. 2015. Overview of the 3rd author profiling task at pan 2015. In *CLEF*, page 2015. sn.

Francisco Rangel, Paolo Rosso, Manuel Montes-y Gómez, Martin Potthast, and Benno Stein. 2018. Overview of the 6th author profiling task at pan 2018: multimodal gender identification in twitter. *Working Notes Papers of the CLEF*, pages 1–38.

Ruder. 2021. Tracking progress in natural language processing.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Matti Wiegmann, Benno Stein, and Martin Potthast. 2019. Overview of the celebrity profiling task at pan 2019. In *CLEF (Working Notes)*.

Xiao Yang, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2017. Transfer learning for multi-language twitter election classification. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 341–348.

Guy or Hombre? Data Augmentation in Conversational Coreference Resolution

Fran Pugelnik, Sara Borzić, Nera Frajlić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{fran.pugelnik, sara.borzic, nera.frajlic}@fer.com

Abstract

Coreference resolution is a challenging task in natural language processing and requires a specifically labeled data set. Labeled textual data is a luxury and often requires human interference. Therefore, instead of collecting more data, proper data augmentation can improve model performance. We test the performance of AllenNLP's coreference resolution model using augmented transcripts from the TV show *Friends*. The idea is to construct additional training instances using synonym replacement. We examine if this augmentation method has a desired boost in performance for coreference resolution task.

1. Introduction

The amount and quality of data greatly affects the performance of any model in any NLP task, including coreference resolution. Instead of collecting more data, it might be more convenient to apply data augmentation to already obtained data to generate additional, synthetic data and make the model generalize better. In this paper, we describe an application of simple data augmentation techniques for improving performance on coreference resolution task. The model chosen for tackling the coreference resolution task is an end-to-end neural model by Lee et al. (2017). Its key idea is to consider all spans in a document as potential mentions to later produce the most likely correct clustering for mentions.

The advantage of using a neural model for coreference resolution is the usage of word embeddings to capture the similarity between words. This can lead to the prediction of false-positive links when the model conflates paraphrasing with kinship or similarity (Lee et al., 2017). We dive deeper into this problem in Section 4. of this paper.

2. Related Work

Manipulations in the input data in NLP tasks often result in system failure, although they would not affect human performance on the same task. Our idea is based on many recent papers that expose the benefits of data augmentation techniques in such cases, and for natural language processing tasks in general. A lot of existing work focuses on using data augmentation for mitigating gender bias in NLP tasks (Zmigrod et al., 2019; Zhao et al., 2018; Lu et al., 2018).

Some similar approaches were explored for improving syntactic parsing (Elkahky et al., 2018), as well as natural language inference (NLI) (Min et al., 2020) and NLI and sentiment analysis (Kaushik et al., 2020). Contextual data augmentation is described in (Wu et al., 2018; Kobayashi, 2018) on various text classification tasks, and other data augmentation operations such as synonym replacement, random insertion, random swap and random deletion are used in (Wei and Zou, 2019) for text classification tasks.

Other interesting techniques, such as back-translation and word replacing with TF-IDF are discussed in (Xie et al., 2019) also for text classification tasks. Augmentation of a word by a contextual mixture of multiple related words is used in (Gao et al., 2019) for the task of machine translation. (Wu et al., 2019) explored the use of existing question answering datasets as data augmentation for coreference resolution. For the task of named entity recognition, various approaches such as label-wise token replacement, synonym replacement, mention replacement, and shuffle within segments are described in (Dai and Adel, 2020). The most natural choice for data augmentation – replacing the words with their synonyms is also used in (Zhang et al., 2015) for controlling generalization error.

Neural models often perform well by using superficial features, rather than more general ones that are preferred and more natural to humans. Data augmentation is used in (Jha et al., 2020) by generating training examples to encourage the model to focus on the strong features.

Considering the previous success of data augmentation techniques on a broad spectrum of topics we think that it might be useful to apply synonym-based augmentation on coreference resolution task in conversational text.

3. Database

The data collection created by Chen and Choi (2016) consists of transcripts of the first two seasons from the TV show *Friends*[1]. Most of the dialogues between the characters are everyday conversations and introduce over 200 speakers. Figure 1 shows an example of a multiparty dialogue present in the dataset. The mention "mom" is not one of the speakers; nonetheless, it refers to the specific person, Judy, that could appear in some other dialogue. Identifying and clustering such mentions might require cross-document coreference resolution.

The dataset follows the CoNLL 2012 Shared Task data format[2]. The data we used consists of documents, each document is delimited and each episode is considered a document. Each word is associated with a word form, part-of-speech tag, and its lemma. A speaker is annotated for each sentence in the training set, as well as in the test set. Each mention is annotated with a belonging entity ID that is con-

[1] https://bit.ly/2SXXn7q
[2] https://bit.ly/3j1mUau

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 65–69, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

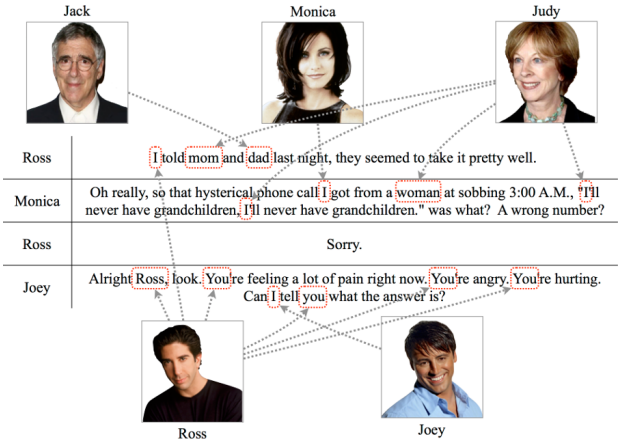


Figure 1: Dialogue example with labeled mentions assigned to entities. Taken from `https://competitions.codalab.org/competitions/17310`

sistent across all documents. For example, Table 1[3] shows a specific sentence from the dataset. The sentence is from the second episode in season 2, the scene number is 0 and the speaker is Chandler Bing. The entity of the mention in this example belongs to their neighbour, the character Ugly Naked Guy.

4. Our Approach

Our experiment consists of an end-to-end neural model by Lee et al. (2017) which will be trained on a semeval 2018 dataset and its augmented counterpart.

4.1. Model's Weaknesses

In the model we used, "The key idea is to directly consider all spans in a document as potential mentions and learn distributions over possible antecedents for each" (Lee et al., 2017). The spans that are not likely to appear in the clusters are discarted. For the spans remaining after this step the model decides whether the span is coreferent with some of the antecedent spans. It returns the resulting coreference links which (after applying transitivity) imply clustering of spans for the given document.

This model uses word embeddings to capture the similarity between words. This can lead to a prediction of false positives when the model confuses paraphrasing with relatedness or similarity. The example given to illustrate this scenario was when the model falsely predicted a link between a pilot and a flight attendant or Prince Charles and his wife Camilla and Charles and Diana.

Another concern is that the model sometimes confuses in rarely occurring patterns and examples that humans wouldn't find challenging.

4.2. Idea

It is assumed that both of the problems explained in 4.1. could be mitigated with a larger corpus of training data which would overcome the sparsity of these patterns.

[3]Some columns were omitted from this preview for clarity

We argue that using data augmentation to extend our corpus would help the model generalize and improve its results on patterns like these while it wouldn't affect the rest of its performance.

For creating a larger dataset we augmented certain words from the existing data set and appended it to the original, which is described in Section 4.3.. One can say that by augmenting the training data with synonyms, we would create duplicate data. However, that is not the case because synonyms from WordNet often don't match the exact synonyms of words from the dataset. As an example, the word 'date', which meant an appointment to meet someone, was in one case replaced with the word 'engagement' which is arguably not a synonym in this context. This is how the noise is added to our corpus and how the better generalization of the model is accomplished.

4.3. Data Augmentation

Our approach consists of using data augmentation techniques for improving AllenNLP's model's performance on coreference resolution task. *nlpaug*[4] is a library for textual augmentation in machine learning experiments. The experiment consists of using the augmenter from WordNet[5] lexical database, which replaces words from the input dataset with its synonyms. More specifically, lemmas of words were replaced with their synonyms. An example of different sentences generated from the same sentence using synonym augmentation is presented in Table 2.

Lemmas that were excluded from the synonym replacement task are stopwords from Nltk[6] (Natural language toolkit). Additionally, to assure that words that greatly affect our task of coreference resolution stay the same, entities (mostly names of the characters) were also excluded from synonym replacement operation. Due to limitations imposed by the .conll file format, it is decided that if a particular synonym does not consist of one word, but is rather a phrase, it will not be a candidate for replacement. The words whose lemmas were replaced by synonyms, had their *Word form* also replaced with the same lemma form. Having that in mind, some information in the created dataset was lost. However, The POS and constituency tags from .conll file were not changed so the replacement of the *Word form* shouldn't cause a lack of too much information. After creating such augmentation, it was simply appended to the existing dataset and presented to the model.

5. Experimental Setup

The pretrained model in AllenNLP library was trained on the English coreference resolution data from the CoNLL-2012 shared task (Pradhan et al., 2012). We tested the model's performance on data described in Section 3., as well as the augmented data described in Section 4.3..

Inspired by Dai and Adel (2020), we simulate a low-resource setting and select the first 16 and 32 episodes from the training set to create the corresponding small and medium training sets to perform data augmentation and ob-

[4]`https://nlpaug.readthedocs.io/en/latest/`
[5]`https://wordnet.princeton.edu/`
[6]`https://www.nltk.org/`

Table 1: Reduced training dataset example.

Document ID	Scene ID	Token ID	Word form	POS tag	Lemma	Speaker	Entity ID
/friends-s01e02	0	0	Ugly	JJ	ugly	Chandler_Bing	(380
/friends-s01e02	0	1	Naked	JJ	naked	Chandler_Bing	-
/friends-s01e02	0	2	Guy	NNP	guy	Chandler_Bing	380)
/friends-s01e02	0	3	got	VBD	get	Chandler_Bing	-
/friends-s01e02	0	4	a	DT	a	Chandler_Bing	-
/friends-s01e02	0	5	Thighmaster	NN	thighmaster	Chandler_Bing	-
/friends-s01e02	0	6	!	.	!	Chandler_Bing	-

Table 2: Data augmentation examples.

ORIG.	Sounds like a date to me.
AUGM.	Sound like a particular date to me. Speech sound like a engagement to me. Sounds comparable a engagement to me.

serve the results. We expect to get the best results by training the model on the largest train set.

For each training set (small, medium, complete) we conduct simple experiments. We split the training set using random seeds on training and validation sets (87.5% - 12.5%). The reason for choosing the unusual train-dev split sizes is because, in this way, an integer division of episodes in the training set is ensured. We then apply data augmentation on the training set and test the model performance on the test set (which is always the same). This experiment is repeated 5 times for each training set size using different random seeds so statistical evaluations could be performed.

We then augmented each of the training sets by adding synonym equivalents for 50% of the training set. The final result were 6 training sets - small, medium, complete, and their corresponding augmented sets.

6. Results

The results are presented in Table 3. The table shows macro-averaged metrics (precision, recall, and F1-score) obtained from testing the model trained on a specific training set (S - small, M - medium, F - full/complete).

6.1. Analysis

As we were unable to perform testing on a large number of training instances (5 runs with different random seeds) due to the time-complexity of the training process, we can't say much about the distribution of test results. However, the observation distribution pairs (e.g. small original - small augmented) are similar in shape, so we ran a non-parametric Mann-Whitney U test in place of an unpaired t-test. We wanted to test whether the observations (specifically F1-scores) in one sample tend to be larger than observations in the other. Considering the fact that the mean results shown in Table 3 fluctuate in different ways, we state specific research hypotheses for each training set size and perform a test calculation at a significance level of 0.05.

For both small and medium data set, the research hypothesis we chose to state is that a randomly selected F1-score obtained from the population of tests conducted on non-augmented data is greater than the same score obtained when testing on augmented data. Informally, after seeing the results presented in Table 3, we decided to argue that the results are significantly better when using non-augmented data. The null hypothesis is rejected in favor of the alternative. Therefore, it appears that this type of data augmentation impairs this models performance.

As for the full data set, according to the mean values of F1-scores, we chose to test the hypothesis that a randomly selected F1-score obtained from the population of tests conducted on non-augmented data differs in any way from F1-score obtained when testing on augmented data. The resulting p-value equals 0.15, therefore the null hypothesis cannot be rejected. We can conclude that no significant difference can be confirmed between the distributions of F1-scores from testing the model on the full data set and the full augmented data set.

6.2. Discussion

The results show that data augmentation with synonyms doesn't improve the performance of the coreference resolution model. Moreover, all subsets of the augmented datasets show lower performance on the test set than the corresponding subsets of non-augmented data. We argue that this is the result of the generalization problem presented in Section 4.1.. Obviously, the polysemy of words has created more problems than benefits for the model. A possible improvement is to use contextual word embeddings in combination with WordNet synonyms for augmenting data so the error due to the lack of context lapses. Another idea would be to include word or span representations that can distinguish between equivalence and paraphrasing. Moreover, some generalization techniques should be applied. In terms of other augmentation methods, antonym replacement or random word generation might help with generalization.

Secondly, we can see that the obtained results have consistently shown that the model achieves a lot lower recall than precision. Knowing that recall denotes how often our model has classified data correctly in regards to the full set of relevant results, we can notice that the model has more problems with recognizing phrases that refer to some entity than it does with correctly deciding on which entity that

Table 3: Results.

	S			M			F	
Data	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
original	0.564	0.431	**0.480**	0.581	0.396	**0.453**	0.579	0.318	0.371
synonym augmented	0.546	0.327	0.395	0.553	0.301	0.356	0.581	0.224	0.285

phrase is referring to. Therefore, it might be interesting to look into the reasons for such phenomena in future work.

7. Conclusion

We presented a simple data augmentation technique for tackling the task of coreference resolution in multiparty dialogues. We showed that synonym replacement for this specific data and model showed no statistically significant improvement in regards to non-augmented data.

Coreference resolution is a challenging NLP problem. The model had problems detecting mentions which can be seen from the low recall scores. While precision scores are better, this model wouldn't be useful in real-world application and human performance on this task is still significantly better.

References

Yu-Hsin Chen and Jinho D. Choi. 2016. Character identification on multiparty conversation: Identifying mentions of characters in TV shows. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 90–100, Los Angeles, September. Association for Computational Linguistics.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2256–2262, Austin, Texas, November. Association for Computational Linguistics.

Xiang Dai and Heike Adel. 2020. An analysis of simple data augmentation for named entity recognition. *CoRR*, abs/2010.11683.

Ali Elkahky, Kellie Webster, Daniel Andor, and Emily Pitler. 2018. A challenge set and methods for noun-verb ambiguity. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2562–2572, Brussels, Belgium, October-November. Association for Computational Linguistics.

Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544, Florence, Italy, July. Association for Computational Linguistics.

Rohan Jha, Charles Lovering, and Ellie Pavlick. 2020. When does data augmentation help generalization in nlp? *CoRR*, abs/2004.15012.

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, New Orleans, Louisiana, June. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark, September. Association for Computational Linguistics.

Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. 2018. Gender bias in neural natural language processing. *CoRR*, abs/1807.11714.

Junghyun Min, R. Thomas McCoy, Dipanjan Das, Emily Pitler, and Tal Linzen. 2020. Syntactic data augmentation increases robustness to inference heuristics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2339–2352, Online, July. Association for Computational Linguistics.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online, July. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea, July. Association for Computational Linguistics.

Jason W. Wei and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. *CoRR*, abs/1901.11196.

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2018. Conditional BERT contextual augmentation. *CoRR*, abs/1812.06705.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2019. Coreference resolution as query-based span prediction. *CoRR*, abs/1911.01746.

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Lu-

ong, and Quoc V. Le. 2019. Unsupervised data augmentation. *CoRR*, abs/1904.12848.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, New Orleans, Louisiana, June. Association for Computational Linguistics.

Ran Zmigrod, Sabrina J. Mielke, Hanna M. Wallach, and Ryan Cotterell. 2019. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. *CoRR*, abs/1906.04571.

Celebrity Profiling

Joško Šestan, Krešimir Bačić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
josko.sestan@fer.hr, kresimir.bacic@fer.hr

Abstract

Social media has gained significant popularity over the last two decades, which partly has to do with celebrities embracing social media websites as easy-to-use platforms for spreading their influence as well as their accessibility to the general public. Celebrities rely on their fan base and social media following in order to grow their popularity and, in turn, their marketing ability. One such social media service is Twitter, a website mainly used to publish short textual posts, along with photos and videos to a lesser extent. Given the large number of celebrities actively using the service, Twitter is great source of data for natural language processing tasks such as predicting various personality traits. In this paper we explored the relevance of features extracted by sentiment analysis in predicting a Twitter user's gender, occupation and level of fame. The experiments were run on the PAN 19 Celebrity Profiling dataset consisting of 33,836 Twitter profiles, each containing 2,181 tweets on average.

1. Introduction

Author profiling is the analysis of a given set of texts in an attempt to uncover various characteristics of the author based on stylistic and content-based features. Increased usage of social media platforms in people's daily lives makes them a valuable source of textual data for author profiling or, in case of this paper, celebrity profiling from tweets. Celebrities post differing content on a daily basis such as paid promotions, business-related announcements and events from their personal lives. In Twitter's case, celebrities can also retweet others' tweets or mention other users in their tweets. Data from social media cannot always be reliable or accurate, as a user could easily provide objectively false information or withhold otherwise relevant facts. Another limitation of using social media posts is the syntactic irregularity of text. Tweets are often full of spelling errors, emoticons, numbers expressed through letters etc., all of which present a significant challenge in any kind of text-based analysis.

In this paper we aim to use features calculated by sentiment analysis as well as generic Twitter NLP features to accurately predict a celebrity's gender, occupation and degree of fame. A task similar to ours was completed as part of the PAN 19 Celebrity Profiling competition (Wiegmann et al.2019). Studying and preprocessing celebrities' tweets allowed us to extract certain features. Using the extracted features we tested how different classifiers perform when used to predict each trait and how their performance changes when different parameter setups are employed.

First, in Section 2 we discuss related work and explain how our paper differs from it. In Section 3 we describe the data set used for training and prediction. Next, Section 4 contains details of data preprocessing, feature extraction and training classification models. Section 5 contains classification results for predicting traits with different classifiers. Finally, our conclusions are presented in Section 6.

2. Related work

Research similar to ours has been done as part of the PAN 19 Celebrity profiling competition. Some authors have tried using sociolinguistic features (Moreno-Sandoval et al.2019) such as the average number of tweets in first person singular form, the average number of tweets in first person plural etc., however to our knowledge using sentiment analysis has not been thoroughly explored. To differentiate from other research on the topic, we have decided to explore the relevance of using sentiment features for training various predictive models.

3. The dataset

As stated in Section 1, the PAN 19 Celebrity Profiling dataset consists of 33,836 Twitter profiles, each containing 2,181 tweets on average. Celebrities are listed as JSON objects, one per line and identified by the id key. The dataset is part of the PAN 19 Celebrity Profiling competition (Wiegmann et al.2019). All possible values for each of the traits are shown in Table 2.

4. Process description

Tweets are usually highly complicated data, containing various different tokens such as mentions, emoticons, URLs and so on. Also, a tweet can be a "retweet" - essentially, another user's tweet shared on the observed user's feed. Considering the previous notes, preprocessing is a necessary step before feature extraction. After the feature extraction process was completed, we proceeded with classification. More information on the classification models used in this research as well as the results of our testing process can be found in Section 5.

4.1. Preprocessing

During preprocessing, we deemed relabeling of hashtags, URLs, emoticons and mentions as necessary. Hashtags were relabeled as _hashtag, URLs as _url, emoticons as _emoji and mentions as _mention. All punctuation marks were removed, as well as stop words and spaces. Also,

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 70–72, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

Table 1: Features description

Feature	Description
avg_hashtags	Average number of hashtags per tweet of a celebrity
avg_mentions	Average number of mentions per tweet of a celebrity
avg_RT	Average number of retweets of a celebrity
avg_url	Average number of URLs per tweet of a celebrity
avg_emoji	Average number of emoticons per tweet of a celebrity
avg_words	Average number of words per tweet of a celebrity
avg_polarity	Average sentiment polarity rate on tweets of a celebrity
avg_subjectivity	Average subjectivity of a celebrity
max_polarity	Maximum sentiment polarity rate on a tweet of a celebrity
min_polarity	Minimum sentiment polarity rate on a tweet of a celebrity
std_polarity	Standard deviation of sentiment polarity rate of a celebrity
std_subjectivity	Standard deviation of subjectivity rate of a celebrity
median_polarity	Median rate of sentiment polarity of a celebrity
median_subjectivity	Median rate of subjectivity of a celebrity

Table 2: Possible values for each of the traits

Trait	Values
fame	rising, star, superstar
occupation	sports, performer, creator, politics, manager, science, professional, religious
gender	male, female, nonbinary

words were saved in lemma form and lowercased. Preprocessing work was done using *spaCy*.[1]

4.2. Feature extraction

Features are built around words, emoticons, hashtags, URLs, mentions and sentiment analysis. Sentiment analysis features were extracted using *spaCy*.[1] As part of sentiment analysis features, polarity and subjectivity of a celebrity's tweets were measured. We thought sentiment analysis might be of great use in predicting the chosen traits as a person's general attitude logically might be influenced by their gender, occupation and fame. For example, a person could have more positive or more subjective tweets depending on what is their occupation. Other features are a somewhat generic Twitter feature set - the average usage of emoticons, hashtags, mentions, number of retweets etc. Features and their descriptions are shown in Table 1.

4.3. Feature fine-tuning

The features' individual relevance was calculated using the kbest function from Python's *scikit-learn*.[2] module. Features were sorted in a descending order based on the relevance scores assigned to them. The feature list was iteratively reduced with the feature deemed least relevant being removed from the list in each iteration. Models from the second round of testing were fitted with reduced training data and their precision scores used in the evaluation of each feature setup. Results have shown each of our features had sufficient relevance to positively influence the prediction.

Table 3: Results from different classifier when predicting Gender

Classifier	Parameters	Score
Nearest Neighbors	weights: distance n: 18	0.7443
Linear SVM	penalty: l1 loss: squared_hinge C: 3.8	0.7331
Decision Tree	depth: 7 features: None	0.7441
Random Forest	depth: 12 features: 4 estimators: 25	0.7575
Neural Network	alpha: 0.05 iters: 1250	0.7630
AdaBoost	estimators: 20 learning rate: 0.7	0.7155

5. Model selection

Various models were fitted with training data and evaluated on the test dataset. The evaluation of models for each of the labels was done in two rounds: in the first, the models were using their default parameter vales. The testing was done with unscaled, standardized and min-max scaled (between -1 and 1) data. All models within 15% precision of the highest performing model made it into the second round. In the second round, the remaining models were tested using several different combinations of parameters specific to the models.

[1] https://spacy.io/

[2] https://scikit-learn.org/stable/

Table 4: Results from different classifier when predicting Occupation

Classifier	Parameters	Score
Nearest Neighbors	weights: distance n: 24	0.5939
Linear SVM	penalty: l1 loss: squared_hinge C: 1.4	0.5711
Decision Tree	depth: 9 features: None	0.5623
Random Forest	depth: 13 features: 6 estimators: 28	0.6151
Neural Network	alpha: 0.1 iters: 1550	0.6315
AdaBoost	estimators: 42 learning rate: 0.45	0.4853

Table 5: Results from different classifier when predicting Degree of fame

Classifier	Parameters	Score
Nearest Neighbors	weights: uniform n: 20	0.7532
Linear SVM	penalty: l1 loss: squared_hinge C: 0.4	0.7516
Decision Tree	depth: 4 features: 5	0.7529
Random Forest	depth: 8 features: None estimators: 15	0.7582
Neural Network	alpha: 0.1 iters: 950	0.759
AdaBoost	estimators: 26 learning rate: 0.645	0.7531

Testing was done on the following predictive models: nearest neighbors, linear SVM, decision tree, random forest, multi-layer perceptron, adaptive boosting and naive Bayes. The precision score was calculated as a mean of 10 tries. The only model with subpar performance was the naive Bayes model when predicting the gender and fame labels. The best results were achieved using data scaled by *spaCy*.[1]'s standard scaler.

In the second round of testing optimal parameter combinations were tested for each of the remaining models. The multi-layer perceptron exhibited the best performance in predicting all of the labels' values, with 76.296% precision when predicting gender, 63.145% for occupation and 75.898% for the fame label. Optimal parameters for each model in each category are in shown in Table 5.

6. Conclusion

The precision achieved by the multi-layer perceptron model, while not state-of-the-art, is high enough to indicate that features calculated by sentiment analysis can be relevant for celebrity profiling. Given a longer time frame, more sentiment features could be implemented and a more satisfactory performance could be achieved. The possibility of chain classification could also be explored, with predictions of one label being used as an additional feature for predicting the next. With a higher base performance, this type of system could produce better results as a correlation between occupation and fame could logically exist.

References

Luis Gabriel Moreno-Sandoval, Edwin Puertas, Flor Miriam Plaza del Arco, Alexandra Pomares-Quimbaya, Jorge Andres Alvarado-Valencia, and L. Alfonso Ureña-López. 2019. Celebrity profiling on twitter using sociolinguistic features.

Matti Wiegmann, Benno Stein, and Martin Potthast. 2019. Celebrity profiling.

Irony Detection Using LSTM/GRU Ensemble With Different Self-attention Activation Functions

Filip Wolf, Filip Prevendar

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{filip.wolf,filip.prevendar}@fer.hr

Abstract

Irony detection is an important task in natural language processing that can aid in sentiment analysis and understanding of spoken and written human language. It is also a very difficult task relying on the nuance of human communication. The SemEval-2018 Task 3 competition saw a plethora of work done in this area and we further explore some of the submitted models. We implement a neural network model with LSTM cells and a self-attention mechanism using dense word embeddings to solve both binary irony classification and multi class classification. We use an ensemble of different attention mechanism activation functions, different numbers of layers and different RNN cells. Our results show that none of these methods extract varying sentence information and thus are not suitable for using in ensemble models.

1. Introduction

Irony has always been an interesting phenomena in human communication and notoriously difficult to understand by machines. It can be described in a manner similar to sarcasm, i. e. saying the opposite of what you mean, although later we describe different types of irony. Irony is often considered a staple of human language and practically unavoidable to normal human communication. Understanding it is therefore a key for systems who want to truly grasp the nuances of human language. In addition, the study of irony can aid us in understanding human sentiment in written and spoken language.

Here we focus specifically on irony detection in Twitter messages as they offer a large, easy to mine corpus of sentences with much work already previously done in this area (Van Hee et al., 2018). That being said, it also poses a challenge, as we need to deal with colloquial language and poorly written sentences. A specific positive of Twitter is that hashtags and similar message markers can be mined for obvious clues of irony and other phenomena. The model we focus on does not use these markers to extract irony information, but they are used for annotators to decide on the types of irony used in the tweets.

We describe two tasks for our model. The first one is binary irony detection. A tweet is either labeled as expressing or not expressing irony. The second task is concerned with classifying tweets into four distinct groups: 1) no irony, 2) irony by polarity contrast, 3) situational irony and 4) other irony. **Irony by polarity contrast** can be described as having an intended meaning opposite to the literal one. For instance, in the sentence `I really love this year's summer; weeks and weeks of awful weather` we can see that the user expresses a positive emotion (`love`), but it's clearly followed up with something negative (`awful`). **Situational irony** is irony in the more traditional sense: `Most of us didn't focus in the #ADHD lecture. #irony`. Lastly, **other irony** is irony that fits in neither of the two previously defined categories, but is nevertheless still ironic: `@someuser Yeah keeping cricket clean, that's what he wants #Sarcasm`. Some would label this as sarcasm.

The SemEval-2018 Task 3 competition saw a great deal of work done in this area. In recent years, neural models, and specifically LSTM networks have greatly improved performance in this and similar tasks. However, most of the submitted models used a large amount of hand-crafted features and sophisticated ensemble models. What caught our attention was the NTUA-SLP (Baziotis et al., 2018) submission as it was truly end-to-end and used a self-attention mechanism, while also showing great results. We thus further explore different attention based RNN/LSTM/GRU ensemble models and report our results.

In section 2. we go over previous work on this topic. Section 3. presents and describes the model we used. Section 4. discusses the results, and finally, in section 5. we give our conclusion.

2. Related Work

In the SemEval-2018 competition (Van Hee et al., 2018), Task 3, 43 teams had submitted their models for the binary classification task, while for multiclass classification, there were models submitted by 31 teams. Here we list some of the more interesting ones.

In (Baziotis et al., 2018), authors used an end to end system which consisted of a word embedding layer, recurrent neural network layer and an attention layer. They used an ensemble of two such networks where one used word embeddings and the other character embeddings. For the RNN layer, they used bi-directional long short-term memory units (LSTMs). They achieved an F_1 measure of 0.6719 for binary classification, and 0.4959 for multiclass classification.

In (Vu et al., 2018) a multi layer perceptron model was used with many handcrafted features and it achieved an F_1 score of 0.6476 and 0.4437 for binary and multiclass classification respectively.

Authors in (Rohanian et al., 2018) separate each tweet

Text Analysis and Retrieval 2021: Course Project Reports (TAR 2021), pages 73–76, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2021.

into two parts and extract hand-crafted features that symbolize contrast between these two parts. In each part, they then do sentiment analysis and combine all the features into a voting classifier with SVM and logistic regression. Their respective F_1 score was 0.6500 for binary and 0.4153 for multiclass classification.

In (Wu et al., 2018), authors used a densely connected LSTM network with multi-task learning. They used one network for predicting whether or not a given tweet had `#irony`, `#sarcasam` or a `#not` hashtag and whether the tweet was ironic or not along with its irony class. They achieved an F_1 measure of 0.705, and 0.495 for binary and multiclass classification respectively.

Authors in (Ghosh and Veale, 2018) used a network called a siamese network. They separated each tweet into two parts and fed each part into a fully connected LSTM network. They then used the output of the LSTM networks as input to a fully connected deep neural network, and added softmax classification at the end. They reached an F_1 score of 0.7234 for binary and 0.5074 for multiclass classification.

3. Our Model

3.1. Dataset

We use the dataset previously collected and provided by the team behind NTUA-SLP (Baziotis et al., 2018). It is composed of 550 million archived Twitter messages from April 2014 to June 2017.

For dense vector representations that capture semantic and syntactic meaning, we use the *word2vec* algorithm with the skip-gram model, a negative sampling value of 5 and minimum word count of 20. (Same as NTUA-SLP). The result is a vocabulary containing 800 000 words with 310-dimensional word embeddings with emojis.

3.2. Preprocessing

Again borrowing from NTUA-SLP, we utilize *ekphrasis*[1](Baziotis et al., 2017), a tool used for tweet tokenization, spell correction, word normalization, hashtag splitting and word annotation.

The first step is **tokenization**. In Twitter, this task is quite challenging, but the tool we use is specifically designed for this and solves the task successfully. The next step is **normalization** where we do spell correction, word normalization and segmentation. Some tokens are also modified by surrounding them with special characters, such as URLs and emails, while some are simply omitted. For spell correction and word segmentation, the Viterbi algorithm is used.

3.3. RNNs

There are three main different types of Recurrent Neural Networks. The generic RNNs, although revolutionary at their time, suffer from exploding and vanishing gradients (Pascanu et al., 2012) which in practice translates to a short memory span in sentences. In order to combat this, LSTMs (Hochreiter and Schmidhuber, 1997) were introduced, which solve this by separating memory and output responsibilities thus allowing gradients to flow over longer distances in word sequences. Specifically, we use bi-directional LSTM (Schuster and Paliwal, 1997) cells for capturing both forward and backward sentence information. In addition, we also use dropout layers between the LSTM layers with a dropout value of 0.1 for regularization.

Aside from LSTMs, we also use GRUs (Cho et al., 2014). They are similar in manner to LSTMs, but simpler in design and use fewer parameters. One of our tasks was testing which one of these two works better for the two tasks previously described and if they extract different sentence information, thereby making them viable for ensemble training. Finally, we compare these two to regular RNNs.

Table 1: Results of the model with different recurrent neural network types and number of layers trained with word embeddings on task A.

RNN cell	layers	attention	F_1
RNN	1	no	0.6867
RNN	2	no	0.7008
RNN	4	no	0.7011
GRU	1	no	0.7360
GRU	2	no	0.7342
GRU	4	no	0.7512
LSTM	1	no	0.7528
LSTM	2	no	0.7443
LSTM	4	no	0.7587
RNN	1	yes	0.7101
RNN	2	yes	0.7030
RNN	4	yes	0.6950
GRU	1	yes	0.7617
GRU	2	yes	0.7587
GRU	4	yes	0.7621
LSTM	1	yes	0.7646
LSTM	2	yes	0.7724
LSTM	4	yes	0.7789

3.4. Self-attention

The attention mechanism has seen increased usage in recent times due to its effectiveness and the recent and influential Google paper (Vaswani et al., 2017). Attention is a simple method to implement that can provide a noticeable boost in performance. Here we test different activation functions with the attention mechanism to see how and if they impact performance. We use a simple linear attention mechanism and cycle through ReLU, tanh and sigmoid activation functions (we report only the latter two). Due to variable batch lengths, we are unable to use more sophisticated self-attention mechanisms. We also try to ensemble them to see if they extract different information from sentences. The self-attention mechanism is defined by:

$$e_i = activation(W_h h_i + b_h)$$

$$a_i = \frac{\exp(e_i)}{\sum_{t=1}^{T} \exp(e_t)}$$

[1] `https://github.com/cbaziotis/ekphrasis`

Table 2: Results of the model with different recurrent neural network types trained with word embeddings with 2 layers on task B.

RNN cell	attention	F_1
RNN	no	0.4472
GRU	no	0.5152
LSTM	no	0.5020
RNN	yes	0.4853
GRU	yes	0.5165
LSTM	yes	0.5172

Table 3: Results while using 2 layers of LSTM units with different attention types and their ensemble for both task A and B.

Attention activation	F_1 for task A	F_1 for task B
tanh	0.7752	0.5231
sigmoid	0.7828	0.5222
ensemble	0.6977	0.3996

$$r = \sum_{i=1}^{T} a_i h_i$$

where *activation* is the activation function and *r* is the sum of the weights a_i assigned to words h_i.

3.5. Ensembling

Ensembling is the act of combining outputs from different models to further boost classification performance with the hopes that different classification models capture varying information from the same input data. There are many types of ensemble techniques, such as majority voting and output averaging. Here we use a simple unweighted average ensemble method:

$$p = \underset{c}{\operatorname{argmin}} \frac{1}{C} \sum_{i=1}^{M} \mathbf{p_i}$$

where C is the number of classes and $\mathbf{p_i}$ is the weight vector of the posterior probabilities for model *i*. The result is a vector that we use as posterior probabilities further on. We only ensembled two models at a time, which means M was fixed at 2. For task A (binary classification) and B (multi class classification), C was fixed at 2 and 4 respectively.

4. Results

The results of our experiments are shown in the tables above and below. We trained our models on the train datasets and tested them on the gold datasets provided for the SemEval 2018 task 3 competition. We used macro F_1 scores to calculate performance.

Table 1 shows F_1 scores for binary classification using different types of recurrent neural network cells, different numbers of layers and with or without using an attention mechanism on task A. Each model is trained along with its F_1 score 5 times and the average is displayed. Table 2 shows F_1 measures obtained in the same fashion but for task B. In all further experiments, the number of layers is fixed at 2. As the tables suggest, an attention mechanism significantly improves the performance of classification when using basic RNN cells on task B (*t* test, $P \leq 0.0001$), but not so much when using more sophisticated units such as LSTMs or GRUs, while on task A, the results are opposite (*t* test, $P \leq 0.001$).

Table 4: Results while using 2 layers of LSTM units with tanh attention activation, without attention and using their ensemble for both task A and B.

Attention	F_1 for task A	F_1 for task B
tanh	0.7623	0.5110
none	0.7647	0.5179
ensemble	0.6386	0.4469

Table 5: Results while using 2 layers of LSTM or GRU units with tanh attention activation and their ensemble for both task A and B.

RNN cell	F_1 for task A	F_1 for task B
GRU	0.7737	0.5286
LSTM	0.7506	0.5354
ensemble	0.6865	0.3384

Next, we tried using different types of attention mechanism activation functions using an LSTM network which we also combine into an ensemble. The results are shown in table 3 for both tasks A and B. Different types of attention activation functions did not show a significant difference (*t* test, $P = 0.3084$), while the ensemble performed worse than each model separately.

After that, we tried using a model with LSTM units and with and without an attention mechanism with a tanh activation function, along with their ensemble. Results are shown in table 4. Since we used models with LSTM cells, we did not see any significant difference between models with and without an attention mechanism as stated previously. Furthermore, the ensemble performed worse than individual models.

Finally, we tried using different recurrent neural network cell types: LSTM and GRU and their ensemble. Results are shown in table 5. There appears to be some significance in using GRU rather than LSTM cells (*t* test, $P = 0.0114$ on task A) although it might be due to luck as we did not notice similar improvements elsewhere. Again, the ensemble performed worse than individual models.

It should also be noted that we tried ensembling using word and character embeddings. Out of the 5 runs we tested, only one showed a slight increase in performance when using an ensemble compared to the individual models. This is in accordance with the results reported in

NTUA-SLP.

5. Conclusion

In this paper we attempted to solve the SemEval-2018 task 3 which is irony detection in English tweets for both binary and multi class classification. We based our work on (Baziotis et al., 2018) and tried out different recurrent neural network cell types, cycled through various amounts of layers, multiple self-attention mechanism activation functions and their ensembles.

We have shown that more sophisticated recurrent neural network cells such as LSTMs and GRUs bring an improvement over regular RNN cells. We have also shown that on task B attention helps significantly when using regular RNNs, but no so much when using LSTM or GRU cells. On the other hand, on task A attention has almost no effect on RNN cells, while GRU and LSTM cells show a significant improvement. Moreover, we have shown that different kinds of attention mechanism activation functions do not make a significant difference. Finally, we show that the described models do not extract different information from sentences and are not suitable for ensembling.

Detecting irony and specific types of irony seems to be a difficult task since we achieved a maximum F_1 score of about 0.77 for classifying if something is irony or not (task A), and only 0.53 for classifying different types of irony (task B).

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada, August. Association for Computational Linguistics.

Christos Baziotis, Nikos Athanasiou, Pinelopi Papalampidi, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, and Alexandros Potamianos. 2018. NTUA-SLP at semeval-2018 task 3: Tracking ironic tweets using ensembles of word and character level attentive rnns. *CoRR*, abs/1804.06659.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Aniruddha Ghosh and Tony Veale. 2018. IronyMagnet at SemEval-2018 task 3: A Siamese network for irony detection in social media. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 570–575, New Orleans, Louisiana, June. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80, 12.

Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.

Omid Rohanian, Shiva Taslimipoor, Richard Evans, and Ruslan Mitkov. 2018. WLV at SemEval-2018 task 3: Dissecting tweets in search of irony. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 553–559, New Orleans, Louisiana, June. Association for Computational Linguistics.

Mike Schuster and Kuldip Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45:2673 – 2681, 12.

Cynthia Van Hee, Els Lefever, and Veronique Hoste. 2018. Semeval-2018 task 3 : irony detection in english tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz 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*, volume 30. Curran Associates, Inc.

Thanh Vu, Dat Quoc Nguyen, Xuan-Son Vu, Dai Quoc Nguyen, Michael Catt, and Michael Trenell. 2018. Nihrio at semeval-2018 task 3: A simple and accurate neural network model for irony detection in twitter.

Chuhan Wu, Fangzhao Wu, Sixing Wu, Junxin Liu, Zhigang Yuan, and Yongfeng Huang. 2018. THU_NGN at SemEval-2018 task 3: Tweet irony detection with densely connected LSTM and multi-task learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 51–56, New Orleans, Louisiana, June. Association for Computational Linguistics.

Author Index