Interpretability of Sentence Embeddings in low-resource Languages

Master-Thesis von Daniel Wehner aus Bad Kissingen
Tag der Einreichung:

1. Gutachten: Dr. Steffen Eger
2. Gutachten: Dr. Johannes Daxenberger

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Ubiquitous Knowledge Processing

Interpretability of Sentence Embeddings in low-resource Languages

Vorgelegte Master-Thesis von Daniel Wehner aus Bad Kissingen

1. Gutachten: Dr. Steffen Eger
2. Gutachten: Dr. Johannes Daxenberger

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, Daniel Wehner, die vorliegende Master-Thesis gemäß § 22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

Darmstadt, den 21. Oktober 2019

(Daniel Wehner)

Abstract

Sentence embeddings have become ubiquitous in the field of Natural Language Processing (NLP). Despite the incredible variety of sentence embedding algorithms, it is still not entirely clear what aspects of sentences are captured by such representations. However, an enormous amount of research has already been conducted on the evaluation of sentence embeddings in the English language. Unfortunately, languages different from English are considered less frequently; and if they are taken into account, the multi-lingual setting is often restricted to higher-resource languages like German, French or Spanish. Languages for which significantly less resources are available are on the other hand investigated only rarely. This circumstance motivates to put such low-resource languages more in focus and to analyze whether the results reported for English are in general transferable to these languages as well or whether the results turn out to be entirely different. This work adopts the methodology of *probing tasks* popularized by (Conneau et al., 2018a) and extends this concept to a multi-lingual setting including the following five languages: English, German, Russian, Turkish and Georgian. In this work, seven probing tasks are implemented for Georgian and nine for the remaining four languages. Additionally, the embeddings are evaluated on a set of three *downstream applications*. The results in these two task types are subsequently correlated in order to provide further insight into how predictive probing tasks are for downstream task performance. The correlations turn out to be language-dependent. Results for low-resource languages differ substantially from the ones obtained for English.

The fact that the evaluation setting in the context of this work partially produced results which did not comply with the literature motivated an additional stability analysis. The outcome of this experiment suggests that the usage of large and balanced data sets produces results which are more comparable with the literature. The evaluation on data sets comprising at least 30k instances is recommended since the ordering of the embeddings becomes more stable as a consequence of more training data. The effect of hyper-parameter tuning on the other hand was found to be less significant, but usually has positive effects. The evaluation in the context of this thesis is mostly done on data sets comprising 10k instances which are partially imbalanced. Therefore, the results have to be treated with caution. It is advisable that future work in this domain incorporates the recommendations given above.

Kurzfassung

Längst haben Sentence Embeddings im Forschungsgebiet des Natural Language Processings (NLP) Einzug gehalten. Trotz der überwältigenden Vielzahl an Algorithmen in diesem Bereich ist erstaunlich wenig darüber bekannt, welche Aspekte von Sätzen durch solche Embeddings repräsentiert werden. In den vergangenen Jahren flossen einige Anstrengungen in die Evaluierung vektorieller Repräsentationen von Sätzen, dies jedoch meist ausschließlich für die Englische Sprache. Leider werden andere Sprachen nur selten betrachtet und dann vor allem solche, für die ausreichend Ressourcen zur Verfügung stehen. Beispiele hierfür sind Deutsch, Französisch oder Spanisch. Sprachen, für die derartig viele Ressourcen nicht vorhanden sind, werden zumeist vernachlässigt. Dieser Umstand motiviert dazu, solche Sprachen verstärkt in den Fokus der Forschung zu rücken. Analysen können Aufschluss darüber geben, ob Muster, die für die Englische Sprache entdeckt wurden, auch für andere Sprachen gelten, oder, ob sich die Ergebnisse gänzlich anders darbieten. Die vorliegende Arbeit bedient sich des Konzepts sogenannter *Probing Tasks*, deren Benutzung vor allem von (Conneau et al., 2018a) vorangetrieben wurde. In diesem Zusammenhang wird diese Form der Evaluierung auf mehrere Sprachen erweitert: Englisch, Deutsch, Russisch, Türkisch und Georgisch. Hierbei werden sieben solcher Klassifikationsprobleme für Georgisch implementiert und neun für die restlichen vier Sprachen. Zusätzlich hierzu werden die Embeddings auf drei sogenannten *Downstream Applikationen* getestet. Die Ergebnisse in diesen beiden verschiedenartigen Aufgabentypen werden nachfolgend korreliert. Die Korrelationsanalyse deckt auf, inwiefern sich das Abschneiden von Embeddings in Downstream Applikationen anhand der Ergebnisse auf den Probing Tasks vorhersagen lässt. Wie sich herausstellt, sind die Korrelationen sprachabhängig, wobei die Ergebnisse in anderen Sprachen teils stark von denen in Englisch abweichen.

Der Aufbau der Experimente im Rahmen dieser Arbeit erzeugte Ergebnisse, die zum Teil nicht mit denen aus der Literatur übereinstimmen. Dies gab Anlass dazu, eine zusätzliche Stabilitätsanalyse durchzuführen. Die Ergebnisse dieser Analyse legen nahe, dass die Benutzung großer und balancierter Datensätze von Vorteil ist. Für die Evaluierung wird deshalb empfohlen, Datensätze zu verwenden, die über mindestens 30k Instanzen verfügen, da die Rangfolge der Embeddings bei Datensätzen dieser Größenordnung deutlich stabiler wird. Der Effekt der Optimierung von Hyperparametern kann demgegenüber als vernachlässigbar eingestuft werden, nichtsdestotrotz ist für gewöhnlich ein leicht positiver Einfluss erkennbar. Die Evaluierung im Rahmen dieser Arbeit wird hauptsächlich auf Datensätzen der Größe 10k durchgeführt, die darüber hinaus teils unbalanciert sind. Aus diesem Grund haben die Ergebnisse dieser Arbeit vorläufigen Charakter und sollten weiter validiert werden. Für zukünftige Forschung in diesem Gebiet ist es daher ratsam, die oben gegebenen Empfehlungen zu berücksichtigen.

Acronyms

AMT	Amazon Mechanical Turk
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long Short-Term Memory
BOREP	Bag of random Embedding Projections
BOV	Bag of Vectors
BOW	Bag of Words
BPE	Byte Pair Encoding
CBOW	Continuous Bag of Words
CNN	Convolutional Neural Network
DSGVO	Datenschutz-Grundverordnung
ELMo	Embeddings from Language Models
ESN	Echo State Network
GDPR	General Data Protection Regulation
GEM	Geometric Embeddings
GNC	Georgian National Corpus
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
LASER	Language-Agnostic Sentence Representations
LSTM	Long Short-Term Memory
MLM	Masked Language Model
MLP	Multi-Layer Perceptron
MT	Machine Translation
NER	Named Entity Recognition
NLI	Natural Language Inference
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
NMT	Neural machine translation
OOV	Out of Vocabulary
PCA	Principal Component Analysis
PoS	Part-of-Speech
RNN	Recurrent Neural Network
SBERT	Sentence BERT

SIF	Smooth Inverse Frequency
SNLI	Stanford Natural Language Inference
SOMO	Semantic Odd Man Out
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SWEM	Simple Word Embedding based Model
TPU	Tensor Processing Unit
TREC	Text Retrieval Conference
UD	Universal Dependencies
UKP	Ubiquitous Knowledge Processing
USE	Universal Sentence Embedding
XML	Extensible Markup Language

Symbols

$\mathscr{B}$	data batch
c	index of center word into cocabulary $\mathscr{V}$
d	number of (embedding) dimensions
$\mathscr{D}$	data set
$\mathfrak{D}$	downstream task
$\boldsymbol{E}$	word embedding matrix
$\mathbb{E}$	embedding space
$\boldsymbol{h}$	encoder output
$\mathscr{J}(\boldsymbol{\theta})$	loss function
m	window size
o	index of context word into vocabulary $\mathscr{V}$
$\mathfrak{P}$	probing task
$\mathbb{R}$	set of real numbers
r_p	Pearson correlation coefficient
r_s	Spearman correlation coefficient
$rank(\bullet)$	rank of a number $\bullet$ in a list of numbers
s_i	i-th sentence of corpus
$\boldsymbol{S}$	embedding matrix of a sentence
T	length of a sentence / number of words in the corpus
$\boldsymbol{v}_s$	sentence embedding
$\boldsymbol{v}_w$	word embedding
$\widehat{\boldsymbol{v}}$	one-hot vector
$\mathscr{V}$	set of (unique) vocabulary words
δ_{ant}	hyper-parameter of *Attract-Repel* (antonymy)
δ_{syn}	hyper-parameter of *Attract-Repel* (synonymy)
$\boldsymbol{\theta}$	learnable parameters of a machine learning model
τ	hinge loss
$\bullet^{\mathsf{T}}$	transpose of a vector or a matrix
$\oplus$ / $[\bullet, \bullet]$	concatenation operator
$\odot$	Hadamard/point-wise product
$\|\| \bullet \|\|$	norm of a vector

Table of Contents

List of Figures

List of Tables

1 Introduction

Current research in artificial intelligence increasingly focuses on better interpretability of machine learning models. Recently ratified laws such as the European **General Data Protection Regulation (GDPR)**[1] dictate that all decisions automatically made by computers have to be justified, i. e. manufacturers who employ machine learning systems must be able to give reasons for why certain decision were taken.[2] This poses a challenging problem, since many algorithms in machine learning are so-called **'black-box' models**. Such a model is usually complicated in such a manner that it is impossible to analyze all the effects that parameters and variables inside the model have on each other.[3] The most prominent example of such a black-box model is the neural network algorithm which is capable of successfully dealing with vast amounts of data. The fact that neural architectures are gradually replacing classical methods which were often more interpretable exacerbates the problem even further. In the domain of Natural Language Processing (NLP), workshops like **'black-box NLP'**[4] attempt to shed light onto this ('open the black box')[5] and to provide explanations.

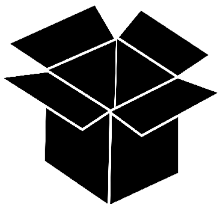


Figure 1: Workshops like black-box NLP try to open the black box and improve the interpretability of neural networks.

Understanding and explaining the output of machine learning models necessitates that the input to these models is understood as well. One of the reasons for bad training or testing performance is the quality of the input features. If the features chosen for the task to be solved are not suitable or not indicative of the class label, even the best model architecture is useless. When it comes to designing machine learning systems involving natural language input, one quickly encounters a concept called **language embeddings**. An embedding is a vectorial representation of natural language in a given granularity, be it single words, sentences, paragraphs or entire documents. Sentence embeddings are vital in NLP because textual input cannot be dealt with by most machine learning algorithms. Despite the fruitful usage of sentence embeddings in a wide range of applications, recent research has brought to light that it is not at all clear what linguistic aspects of sentences are captured by these encodings (Conneau et al., 2018a, inter alia). But exactly this knowledge is invaluable when choosing the right encoder architecture for a specific task at hand. Each task is different and requires that other facets of sentences be represented in order to obtain useful predictions. Very often, **probing** as well as **downstream tasks** are used to assess the representational power of sentence embedding architectures (Conneau and Kiela, 2018, inter alia). This approach is also adopted in this thesis.

Research Questions. Most of the research with respect to the evaluation of sentence embeddings is restricted to the English language. **This thesis therefore aims to extend the focus to a multi-lingual setting by also considering lower-resource languages.** For this purpose, the languages **Russian**, **Turkish** and **Georgian** were chosen. We also conduct the analysis for **English** and **German** to compare the results to higher-resource languages as well as to the literature. *This work is centered around the following research questions:* ❶ One of the major goals is to provide an analysis which addresses the question in how far the language chosen influences the performance of state-of-the-art sentence embedding techniques. Embeddings working well in English may not do so in other languages. ❷ A review of the literature further reveals that the task settings to evaluate the embeddings are not standardized and as a consequence differ from publication to publication. This is why a stability analysis is performed by varying the following factors: `language`, data set `size`, `classifier` architecture, `class balance` and `hyper-parameter optimization`. This way it is possible to draw conclusions about the stability of the relative embedding ordering.

Contributions. In a first step, this document provides an overview of currently used state-of-the-art embedding algorithms. The encoders were selected such that a wide range of different techniques is included in the analysis. Most of the time pre-trained models are available only for English, i. e. that models have to be trained from scratch for the other languages. We trained *sent2vec*, *Quick-Thought* and *InferSent* sentence embeddings as well as *word2vec* word embeddings which form the basis for some of the sentence embedding techniques. To conduct the evaluation, it was furthermore required to setup probing and downstream tasks for each target language. For this purpose, we searched special tree banks and other corpora. In order to overcome the lack of data, especially for Georgian, it was necessary to translate

[1] The regulation is called 'Datenschutz-Grundverordnung (DSGVO)' in German.
[2] Cf. https://medium.com/pachyderm-data/hold-your-machine-learning-and-ai-models-accountable-de887177174c (retrieved: October 07, 2019)
[3] Cf. https://towardsdatascience.com/the-black-box-metaphor-in-machine-learning-4e57a3a1d2b0 (retrieved: October 07, 2019)
[4] Cf. https://blackboxnlp.github.io/ (retrieved: September 04, 2019)
[5] Image source: https://blog.f1000.com/wp-content/uploads/2017/06/black_box_blog.png (retrieved: September 04, 2019)

some of the existing English corpora or to create new data sets from scratch. Our evaluation comprises probing as well as downstream tasks. We present the results on these two task types and successively compute correlations between them. Based on the results of the stability analysis, recommendations are given for future work and research. The following list summarizes the main contributions of this thesis in a chronological order and also serves as the guiding line for the entire project:

1. **Review** of modern word and sentence embedding algorithms.

2. **Acquisition of word and sentence embedding models** for the five languages English, German, Russian, Turkish and Georgian, either by downloading pre-trained models or by training the models from scratch (namely *word2vec*, *sent2vec*, *Quick-Thought* and *InferSent*).

3. **Creation of a set of probing tasks** guided by an analysis of the characteristics of each language (not all probing tasks make sense in all languages, e. g. testing for gender information in Turkish is not sensible, since this concept is not present in this language).

4. **Creation of downstream task data sets**, e. g. an annotated sentiment data set for the Georgian language based on data extracted from Georgian Twitter.

5. **Translation** of downstream task data sets and other corpora, e. g.:
   - Stanford Natural Language Inference (SNLI) corpus (partially translated due to server-side constraints)
   - Ubiquitous Knowledge Processing (UKP) sentential argument mining corpus
   - TREC question type data set

6. **Evaluation of sentence embeddings** in all five languages followed by a discussion of the results. A **stability analysis** is furthermore performed for a selected probing task with respect to the factors:
   - `Language`
   - Data set `size`
   - `Classifier` architecture
   - `Class balance`
   - `Hyper-parameter optimization`

7. **Recommendations** for future work and research.

Organization of the thesis. This thesis is organized as follows: Chapter 2 on the following page first of all introduces related work which represents the foundation of this thesis. The following chapter 3 on page 7 introduces the concept of word embeddings which is essential for many sentence-level algorithms. Three of the most common techniques in this domain are presented which are also used in the context of this work: *word2vec*, *FastText* and *Attract-Repel*. Subsequently, chapter 4 on page 12 introduces the major sentence embedding approaches used in modern applications. All of the presented embeddings are subject to the evaluations conducted at a later stage of the present thesis.

After that, chapter 5 on page 21 provides detailed information about the probing tasks: The chapter begins by introducing probing tasks which were already used in the literature. Subsequently, the low-resource target languages are briefly outlined in order to enable a proper assessment of the implementability of the probing tasks in these languages. These explanations are followed by details about the selection of probing tasks and how the data sets were created. Finally, the classifier architecture employed in the probing task setup is presented. Analogously, chapter 6 on page 41 gives information about the set of downstream applications taken into account. The evaluation results, the discussion as well as the recommendations for future work can be found in chapter 7 on page 46, before the final chapter 8 on page 69 summarizes the main results and concludes this document.

2 Related Work

2.1 Introduction

This chapter introduces scientific research which forms the basis for this thesis. Section 2.2 first of all demonstrates which efforts have already been devoted into the creation of monolingual and multilingual word and sentence embedding algorithms. The focus of the subsequent section 2.3 on the following page is on relevant literature involving the evaluation of sentence embeddings. Overall, this chapter is designed to give a general overview of all relevant concepts without explaining them in-depth. If necessary, a more detailed explanation is given in the respective chapters where the concepts are needed.

2.2 Vectorial Representations of natural Language

Word embeddings. State-of-the-art NLP applications rely on vectorial language representations usually generated by neural networks. The idea to learn representations of natural language – commonly known as **embeddings** – by leveraging such neural architectures is by no means novel. One of the first approaches dates back to the year 2003 when (Bengio et al., 2003) introduced the neural language model which was devised to mitigate the **curse of dimensionality** when learning the joint probability function of word sequences. This could be achieved *'by learning a **distributed representation for words**'*. The major breakthrough, however, was achieved approximately ten years later. Back then, (Mikolov et al., 2013a) proposed the famous *word2vec* algorithm which comes in its two well-known flavors, *Skip-Gram* and *Continuous Bag of Words (CBOW)*. Only one year later, (Le and Mikolov, 2014) published another paper containing several extensions to the *Skip-Gram* model. Two methods, namely **negative sampling** and **sub-sampling of frequent words** were developed to increase the robustness of the resulting word embeddings.

In the following years, many were to join this line of research and new approaches to embedding natural language were invented. Among them is a word embedding called *GloVe* proposed by (Pennington et al., 2014). In contrast to *word2vec*, this algorithm also takes **global corpus statistics** into account to improve the representations learned. Another example is *FastText*, an algorithm which includes **sub-word information** by averaging character *n*-grams (Bojanowski et al., 2017). This model is perfectly suited to handle Out of Vocabulary (OOV) words. (Mrkšić et al., 2017) further propose to fine-tune word embeddings by leveraging **linguistic constraints** like synonymy/antonymy relationships. This idea gives rise to an algorithm called *Attract-Repel*. More recently, **contextualized word embeddings** have emerged. Examples include *Bidirectional Encoder Representations from Transformers (BERT)* (Devlin et al., 2018) which builds on the transformer architecture (Vaswani et al., 2017) and *Embeddings from Language Models (ELMo)* (Peters et al., 2018). Such embeddings assign word representations based on the surrounding words. This way it is possible to model **polysemous words**, i. e. words that can have different meanings in different contexts.

Sentence embeddings. Soon the question arose whether it is feasible to embed even larger spans of text that go beyond single words. The notion of sentence embeddings was born. (Yang et al., 2018) differentiate between **non-parametric** and **parametric** algorithms, where techniques of the first kind (a. k. a. compositional models) are quite frequently applied and often serve as a baseline for more elaborate ones. In this context, each word in the sentence is represented by a word embedding produced by a freely-selectable word embedding algorithm which maps words to vectors. These word embeddings subsequently undergo a pooling procedure to obtain a single sentence representation. Many different pooling strategies have already been investigated in the literature: These range from average-, min-, or max-pooling to more sophisticated strategies like *hierarchical pooling*. (Shen et al., 2018) named these kind of pooling algorithms *Simple Word Embedding based Models (SWEMs)*. Further, (Rücklé et al., 2018) suggest to concatenate various power means to improve the expressiveness of sentence embeddings. Other approaches like *Smooth Inverse Frequency (SIF)* (Arora et al., 2017) or *Geometric Embeddings (GEM)* (Yang et al., 2018) make use of a method called Singular Value Decomposition (SVD)/Principal Component Analysis (PCA) to remove projections on principal components before averaging.

Parametric methods, on the other hand, **train sentence embeddings from scratch**. One of the examples can be regarded as a generalization of the *Skip-Gram* model to the sentence level. Its inventors called the algorithm *Skip-Thought* (Kiros et al., 2015). (Logeswaran and Lee, 2018) reformulate *Skip-Thought*'s objective as a multi-class prediction problem to address several issues of the original model. Other algorithms include *InferSent* (Conneau et al., 2017) which is trained on the *SNLI* data set (Bowman et al., 2015) and *sent2vec* (Pagliardini et al., 2018) which computes sentence embeddings by averaging character *n*-grams like *FastText*, but on sentence level. Recently, (Reimers and Gurevych, 2019) have introduced *Sentence BERT (SBERT)*, a version of BERT which is designed to create sentence embeddings.

Multi-lingual embeddings. Next to monolingual embeddings, it is also possible to train multilingual representations, i. e. embeddings for more than one language that share an embedding space. One of the first approaches was proposed by (Chandar et al., 2013) who trained bi-lingual auto-encoders. (España-Bonet et al., 2017) leverage Machine Translation (MT) architectures to create bilingual embeddings while (Zhou et al., 2016) try to align the representations of documents and their translated counterparts by minimizing the Euclidean distance between the two. More recently, (Artetxe and Schwenk, 2018) introduce *massively multilingual sentence embeddings* which are integrated into the *Language-Agnostic Sentence Representations (LASER)* library.[6][7] The model is trained in a multi-lingual fashion and all languages share a joint embedding space. Thus, the same model can be used for a variety of languages. Currently, 93 languages are supported. *BERT*, too, offers a multilingual version which can be used to obtain embeddings in 104 different languages.[8]

2.3 Evaluation of Word and Sentence Embeddings

While sentence embeddings are nowadays well-established in the NLP community, it is not entirely clear what features of sentences these embeddings encode (Conneau et al., 2018a, inter alia). Sentence representations are **far from being universal** (Perone et al., 2018), i. e. there is no embedding that fits perfectly to all tasks. The knowledge about which linguistic properties of a sentence are learned by individual sentence embedding algorithms is therefore crucial for the choice of the best technique for the problem at hand.

A classical approach frequently applied to evaluate sentence representations is to use them in downstream applications. In such a setting, the quality of the embeddings can be judged with respect to their performance in a real-word scenario. Naturally, this provides useful insights. However, (Conneau et al., 2018a) point out that **such evaluations are encoder-specific and results obtained in one setting are not necessarily predictive of the performance in another setting**. In order to mitigate this shortcoming, they introduced a set of ten so-called *probing tasks* which they define to be ***'classification problem[s] that [focus] on simple linguistic properties of sentences'***. According to the authors, probing tasks have the following advantages over downstream applications:

1. Probing tasks are accompanied by **easier interpretability** due to their simplicity.

2. It is less difficult to **control for biases**. For example (Lai and Hockenmaier, 2014) could show that a classifier can achieve high accuracy scores on the *SICK-E* data set (Marelli et al., 2014) by focusing on explicit negation words. With less complex probing tasks it is much easier to detect such hidden biases and idiosyncrasies of the data set which the model exploits.

3. The concept of probing tasks is **independent of the encoder architecture**. All previously introduced evaluation methods were designed for specific use cases like e. g. MT. As a consequence, it was not straightforward to transfer them to other settings or embedding architectures.

(Conneau et al., 2018a) categorized the tasks into three distinct types: ❶ **Surface information tasks**, ❷ **syntactic tasks** as well as ❸ **semantic tasks**. The authors designed all the probing tasks to fulfill the following requirements:

1. A probing task only takes **single sentence representations as input**. This also excludes feeding additional input like word embeddings into the classifier. This would severely affect the evaluation results in a negative way, since the performance cannot be traced back clearly to the sentence representation alone.

2. The probing tasks setup should allow for the creation of **large training and validation sets**. According to (Conneau et al., 2018a), large amounts of data may be necessary, since the linguistic information might be encoded in a non-linear fashion. Non-linear classifiers are usually characterized by a vast amount of parameters which require more data to be tuned properly.

3. It should be possible to **control for 'nuisance factors' and 'hidden biases'**.

4. The tasks should **'cover an interesting set of linguistic properties'**.

[6] In the following, these embeddings will be referred to as *LASER* embeddings.
[7] Cf. https://github.com/facebookresearch/LASER (retrieved: September 04, 2019)
[8] https://github.com/google-research/bert/blob/master/multilingual.md (retrieved: September 04, 2019)

The work of (Conneau et al., 2018a) was inspired by pioneering efforts made by (Adi et al., 2017; Ettinger et al., 2016; Shi et al., 2016) who were the first to introduce the notion of probing tasks. (Shi et al., 2016) were primarily interested in the question whether MT systems can learn source syntax. In this context they proposed so-called **syntactic labels** to probe sentences for syntactical properties. (Adi et al., 2017), on the other hand, put the focus on testing embeddings for surface information (e. g. the length of a sentence or the containment of words, etc.). (Ettinger et al., 2016) were above all interested in semantic information retained by sentence embeddings. He and his colleagues introduced tasks which make use of **semantic roles** and probed sentences for **entity-event relationships**. Further probing tasks and evaluation methodologies were introduced by (Zhu et al., 2018) who made use of cosine similarities to evaluate sentence embeddings. They constructed triplets of sentences S, S^* and S^+ by having the original sentences undergo **sentence modification schemes** like *'passivization'*, *'synonym substitution'* or *'quantifier negation'*. For instance, they constructed a sentence triplet: (S: *'The fox is jumping over the dog'*, S^*: *'The fox is not jumping over the dog'*, S^+: *'The fox is hopping over the sleeping dog'*). Despite the fact that sentences S and S^* exhibit a quite similar surface form, their meaning is antonymous. On the other hand, S and S^+ use quite different words but have a similar meaning. Hence, (Zhu et al., 2018) claim that the similarity of S and S^* should be smaller than the one between S and S^+: $\text{sim}(S, S^*) < \text{sim}(S, S^+)$. Embeddings complying with this requirement will get a positive test result.

(Gulordava et al., 2018; Linzen et al., 2016) probed Recurrent Neural Network (RNN) representations for long-range dependencies such as subject-verb relations and mainly found that such representations fail to properly represent long-range dependencies. (Naik et al., 2018) created so-called **stress test data sets** to investigate errors of Natural Language Inference (NLI) classifiers. The authors found that if premise and hypothesis sentences have a large word overlap, the model is often inclined to label the sentence pair as an entailment, even if the sentences have no common meaning. If the overlap is small, related sentences are often labeled as neutral. Based on this error analysis, large-scale data sets were constructed aiming at testing models for such phenomena. The authors distinguish between **competence tests**, **distraction tests** and **noise tests**. If the model wants to excel at competence tests, it has to perform complex reasoning. Distraction test data sets contain adversarial examples which aim to elicit a false prediction from the model whereas noise tests give insight into whether the model is robust with respect to spelling errors.

The evaluation of vectorial representations of sentences is often restricted to the English language and only a small number of researchers also consider a multilingual setting. (Cer et al., 2017) for example published semantic similarity data sets in four languages dedicated to evaluation. Other authors released multilingual training and evaluation data which were either produced by translating English resources (Agic and Schluter, 2017; Mehdad et al., 2011) or by additional annotation of parallel corpora (Negri et al., 2011). (Conneau et al., 2018b) created a new English data set similar to the MuliNLI data set (Williams et al., 2018), but translated it into 15 languages, including some low-resource languages like Urdu. (Sahin et al., 2019) use multilingual probing tasks for word embeddings in several languages. They point out that English is rather poor from a morphological point of view and that different results have to be expected in other languages which are more diverse. In fact, they found that correlations between probing and downstream task results tend to be higher in morphologically richer languages. Very recently, (Krasnowska-Kieraś and Wróblewska, 2019) have extended on the ideas of Conneau and colleagues by introducing a slightly modified set of probing tasks for the Polish language. They further enriched Conneau's suite of probing tests with the PASSIVE task which probes for voice information. They mainly found that *LASER* embeddings can be considered superior to other techniques, yet they were unable to give specific reasons for the outstanding performance of this sentence encoder.

(Ravishankar et al., 2019) conduct an evaluation on five different languages (English, French, German, Spanish and Russian) using the set of Conneau's probing tasks with minor modifications. The encoder architecture the authors leverage is very similar to the one used for the *InferSent* model (cf. section 4.3.2 on page 15). The encoder also uses *FastText* word embeddings as a basis.

❶ As a first step, the authors map the *FastText* embeddings for all languages into the English embedding space using the *VecMap* approach introduced by (Artetxe et al., 2016). ❷ Subsequently, they train an English encoder (cf. left-hand side of figure 2 on the following page) on high-quality NLI data. ❸ In the next phase, the authors encode parallel sentences, where the first sentence is given in English and the second one is its translated counterpart in one of the target languages. The English sentence is encoded using the trained English encoder from before (cf. step ❷), whereas the target-language sentence is encoded using a different encoder whose weights are initialized randomly. This encoder is subsequently trained on the mean squared error between its output and the English representation (this step is illustrated in the middle part of figure 2 on the next page). This procedure is repeated for all languages. ❹ After training, the target-language encoder is used in the probing tasks as usual (cf. right-hand side of figure 2 on the following page). The evaluation on probing tasks is then conducted on parallel data the authors extracted from Wikipedia. The encoder mapping procedure was first introduced by (Conneau et al., 2018b).

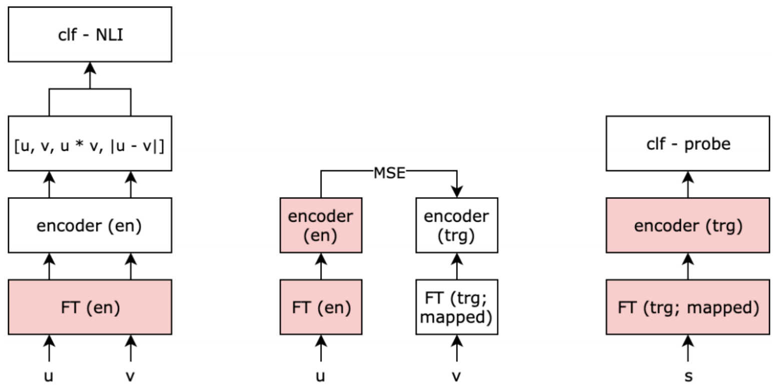


Figure 2: Illustration of the encoder mapping procedure employed by (Ravishankar et al., 2019). The image was taken from (Ravishankar et al., 2019)

(Conneau and Kiela, 2018) furthermore noticed that there was **no standard approach to the evaluation of sentence embeddings**. In each publication, different setups were used such that comparisons between individual results started to lack validity. As a consequence, they developed the *SentEval* framework, an easy-to-use open-source Python project[9] which comes with diverse downstream classification tasks as well as the set of probing tasks introduced by (Conneau et al., 2018a). (Perone et al., 2018) made use of this framework to conduct a comprehensive study in which they evaluated more state-of-the-art embedding architectures. Their experiments confirmed that there is no universal sentence embedding and furthermore showed that a Bag of Vectors (BOV) approach using contextualized *ELMo* embeddings achieves high performance across many probing tasks.

Despite the convenient usage of such frameworks, (Eger et al., 2019) still notice several pitfalls that many researchers frequently tab into when it comes to probing sentence representations. These pitfalls include e. g. **using different classifier architectures** for the evaluation or **comparing sentence embeddings of different sizes**. The latter issue was already realized by (Rücklé et al., 2018) who pointed out that comparing usually low-dimensional average methods to higher-dimensional trained embeddings was unfair and therefore proposed to concatenate several average embeddings to increase their dimensionality. Finally, in order to facilitate a better evaluation, (Eger et al., 2019) conjecture that the set of downstream applications should be enriched with more demanding tasks where simple compositional models and random encoders are outperformed by trained encoder architectures more decisively.

Motivation and Goal of this Thesis

The above explanations show that the number of researchers taking a multi-lingual setting into account is small but increasing steadily. Nevertheless, the set of languages is still often limited to high-resource languages like German, French or Spanish. Overall, much less research could be found that focuses on probing sentence embeddings in lower-resource languages. It is therefore unclear whether the results obtained for high-resource languages are generalizable to languages for which by far less data is available. This circumstance makes it worthwhile to put lower-resource languages more in focus. Providing more insight into sentence embeddings in lower-resource languages is the goal of this thesis.

2.4 Summary

This chapter introduced the foundational work for this thesis. The literature review made clear that a large number of sentence embedding algorithms exists. However, large-scale evaluations of these techniques have long been neglected. What is more, before the release of *SentEval*, no standard framework existed which could be used to assess the quality of the embeddings in a comparable way. Also, the probing task literature is mostly restricted to the English language, but successively, more and more researches are beginning to shift their focus to other languages which are morphologically richer than English. Most of the time only high-resource languages are taken into account, while low-resource languages are omitted. **This is why the present thesis aims to extend the probing task methodology to lower-resource languages in order to investigate whether the patterns found for high-resource languages are still reproducible if by far less resources are available.**

[9] https://github.com/facebookresearch/SentEval (retrieved: September 03, 2019)

3 Word Embedding Algorithms

3.1 Introduction

Word embeddings form the basis for many sentence embedding algorithms. Therefore, this chapter introduces in more detail what word embeddings are, what approaches exist and how these approaches work. Section 3.2 introduces *word2vec*, while the two subsequent sections are dedicated to *FastText* and *Attract-Repel*, respectively (cf. sections 3.3 on the following page and 3.4 on page 9). The last section 3.5 on page 10 finally shows how we acquired word embedding models for later experiments. Available pre-trained vectors were downloaded while others had to be trained from scratch. The section will also provide training details such as which values were used for the hyper-parameters of the respective models.

3.2 Word2Vec

In order to learn dense word representations, (Mikolov et al., 2013a) introduced a neural approach they called *word2vec*. This algorithm was considered a major breakthrough in the NLP community by the time it was published. The learning process of *word2vec* is guided by an **auxiliary task** which is inspired by language modeling (Bengio et al., 2003). In contrast to language models which predict the next word in the sequence given all previous words, *word2vec* either predicts the center word given the context words or the context words given the center word. The first variant is called *CBOW*, while the second one is referred to as *Skip-Gram*. The following paragraphs refer to the *Skip-Gram* architecture, but the concepts are in general transferable to the *CBOW* variant as well. Figure 3 provides a visual illustration of the explanations:

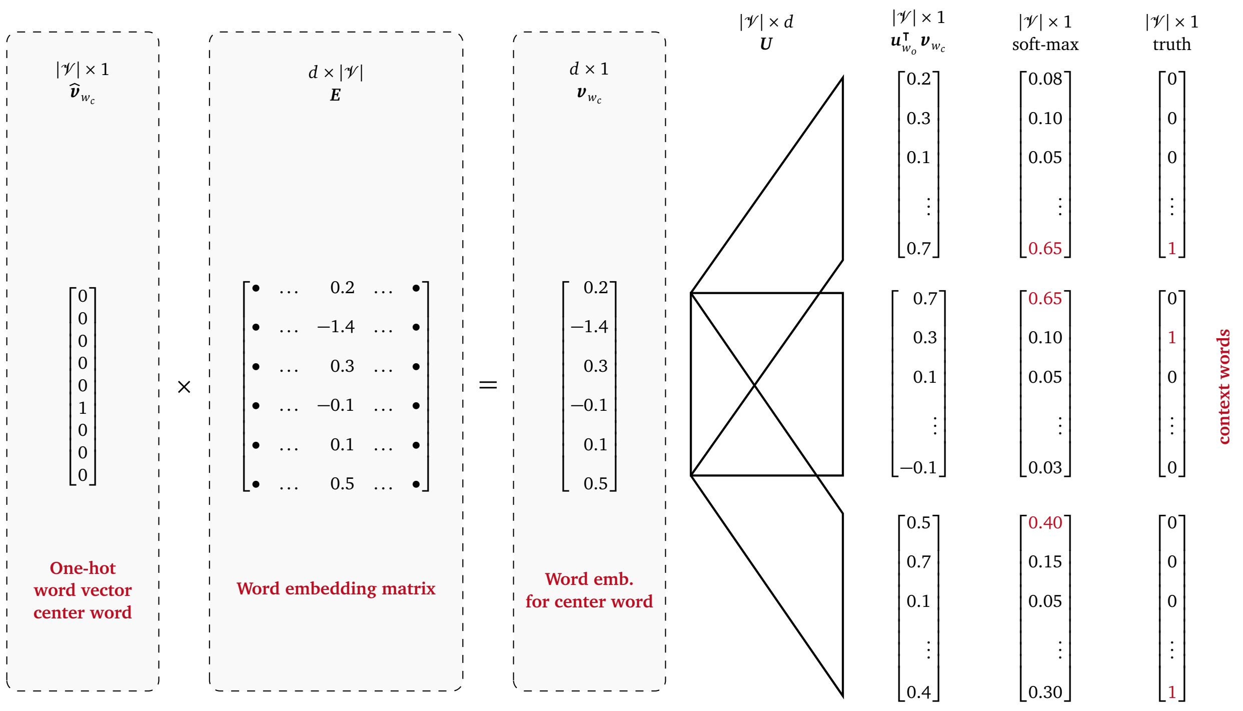


Figure 3: The Skip-gram model in detail. Image taken and adapted from Manning: https://www.youtube.com/watch?v=ERibwqs9p38 (Lecture recording from Stanford university; retrieved: September 04, 2019).

Given the center word $w_c^{(t)}$ at time step t, the *Skip-Gram* model has to predict the surrounding words which are given by a sliding window of size m centered on $w_c^{(t)}$. To this end, $w_c^{(t)}$ which is represented as a one-hot vector $\hat{\boldsymbol{v}}_{w_c^{(t)}} \in \mathbb{R}^{|\mathcal{V}| \times 1}$, is first multiplied by a shared embedding matrix $\boldsymbol{E} \in \mathbb{R}^{d \times |\mathcal{V}|}$, where d is the desired dimensionality of the resulting word embeddings. This matrix contains the weights of the network connecting the input layer to the single hidden layer (with identity activation). The weights in matrix E represent the word embeddings once the model is fully trained. Since the input vector has the one-hot property, the vector-matrix multiplication retrieves one specific column from the matrix, namely the respective (intermediate) embedding of the center word $\boldsymbol{v}_{w_c^{(t)}}$.

The model must be provided with a supervision signal in order to adjust the weights during training. In machine learning, this signal is usually obtained by comparing the predictions of the model with the desired output (supervised learning). This results in a training loss whose value is dependent on how far off the model was with its predictions. Based on the loss, the model parameters are adjusted to a greater or lesser extent. However, the training corpus does not have any dedicated labels in this context. Instead, the words in the corpus themselves are used. This approach can be considered a mixture of supervised and unsupervised learning which is commonly known as **self-supervision**. The labels which are used by *Skip-Gram* are given by the one-hot vectors of the respective context words. To obtain the prediction of the model, one must further multiply the center word embedding by a matrix $\boldsymbol{U} \in \mathbb{R}^{|\mathcal{V}| \times d}$. The resulting vector then serves as input to a soft-max activation which yields a probability distribution over all possible words in the vocabulary $\mathcal{V}$. To be more precise, each word has two representations: One center word representation (contained in matrix $\boldsymbol{E}$) and a context word representation (contained in matrix $\boldsymbol{U}$). Matrix $\boldsymbol{U}$ is needed only for the training procedure and is discarded in the end. During training, the *Skip-Gram* architecture attempts to **maximize the negative log-probability of the context words given the center word**:

$$\mathcal{J}(\boldsymbol{\theta}) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log p(w^{(t+j)} | w_c^{(t)}; \boldsymbol{\theta}), \tag{1}$$

where the probability is given by the soft-max activation:

$$p(o|c) = \frac{\exp\{\boldsymbol{u}_{w_o}^{\mathsf{T}} \boldsymbol{v}_{w_c}\}}{\sum_{\ell=1}^{|\mathcal{V}|} \exp\{\boldsymbol{u}_{w_\ell}^{\mathsf{T}} \boldsymbol{v}_{w_c}\}}. \tag{2}$$

Vector $\boldsymbol{\theta}$ denotes the parameters of the model. In this case these are the entries of the matrices $\boldsymbol{E}$ and $\boldsymbol{U}$. T is the number of words in the training corpus whereas t represents the index into the training corpus. The latter must not be confused with o (context) and c (center) which are indices into the vocabulary $\mathcal{V}$. $\boldsymbol{u}_{w_o}^{\mathsf{T}} \boldsymbol{v}_{w_c}$ is called the scoring function $s(c, o)$.

One property of *word2vec* is that the learned embeddings do not only capture **syntactical features** of natural language, but incorporate **semantic aspects** of words as well. (Mikolov et al., 2013b) detected this desirable property by defining nine syntactic and five semantic tasks/questions, e. g. `Adjective-Adverb` (syntactic) or `Man-Woman` (semantic). The task is to answer questions like *'What is the word that is related to king in the same sense as man is related to woman?'*. The authors were able to demonstrate that such questions can be answered by performing simple algebraic operations on the corresponding word vectors: $\boldsymbol{v}_{king} - \boldsymbol{v}_{man} + \boldsymbol{v}_{woman} \approx \boldsymbol{v}_{queen}$ (cf. figure 4).

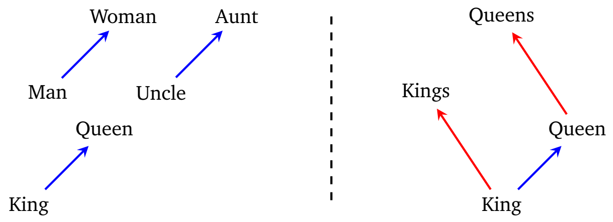


Figure 4: Similarity tasks can be solved by performing simple algebraic vector operations. The image was taken and adapted from (Mikolov et al., 2013b).

3.3 FastText

Despite the great impact *word2vec* had, (Bojanowski et al., 2017) could identify drawbacks. The authors observed that **the algorithm treats words as atomic units**. This design choice entails problems for words not contained in the training corpus. A large amount of **OOV words** is the consequence. It is particularly difficult to learn robust word embeddings for morphologically rich languages like e. g. Turkish. In order to address this issue, (Bojanowski et al., 2017) **enrich the *Skip-Gram* model with sub-word information, where words are considered bags of character n-grams**. The word embedding is then defined as the sum of the respective n-gram embeddings. Given that the training corpus is sufficiently large, this method basically avoids any OOV words. If a word did not exist in the training data, its embedding can nevertheless be obtained by summing the respective n-gram embeddings. (Bojanowski et al., 2017) were inspired by (Schütze, 1993) and (Wieting et al., 2016a) who introduced similar concepts.

In order to formalize this idea, let w denote an arbitrary word in the vocabulary $\mathscr{V}$. $\mathscr{G}_w$ denotes the set of n-grams the word w consists of. A dedicated vector representation $\boldsymbol{z}$ is learned for each n-gram appearing in the training corpus. (Bojanowski et al., 2017) subsequently replace the original scoring function of *word2vec* by the sum of all character n-grams of w:

$$s(c,o) = \sum_{g \in \mathscr{G}_{w_c}} \boldsymbol{z}_g^{\mathsf{T}} \boldsymbol{u}_{w_o} \tag{3}$$

The inventors of *FastText* suggest that n be chosen to be in the range from 3 to 6. The authors present an example: Let n be set to 3. The n-grams of the word *'where'* are given by: `<wh`, `whe`, `her`, `ere`, `re>`. Before the creation of the n-grams a word is enclosed in angle brackets (`<...>`).

3.4 Attract-Repel

(Mrkšić et al., 2017) suggest to incorporate linguistic constraints into the learning process to improve the quality of the word embeddings. Examples for such constraints are **synonymy/antonymy relationships** between words. Synonymy constraints are introduced to reduce the distance between embeddings representing words with a similar meaning (*attract*) while antonymy relationships allow for pushing vectors further away from each other if the words they represent have contradictory meanings (*repel*). Such relationships can be obtained from lexical resources like e. g. *WordNet* (Miller, 1995). In general, two approaches exist how such additional constraints can be incorporated into the process of embedding creation: ❶ **By training word vectors from scratch while respecting the linguistic constraints** or ❷ **by reusing existing word representations which are fine-tuned such that the embeddings obey these constraints**. The authors call this procedure **semantic specialization**. *Attract-Repel* implements the latter approach and was inspired by techniques called *retro-fitting* (Faruqui et al., 2015), *Paragram* (Wieting et al., 2015) and *counter-fitting* (Mrkšić et al., 2016).

The inventors of *Attract-Repel* define S and A to be the sets of synonymous and antonymous word pairs, respectively. $(\boldsymbol{x}_l, \boldsymbol{x}_r)$ describes such a pair, where the words are given by their corresponding vectorial representations. In each training iteration, one mini-batch $\mathscr{B}$ is used consisting of k_1 synonymous word pairs ($\mathscr{B}_S$) and k_2 antonymous word pairs ($\mathscr{B}_A$). Let further denote $\mathscr{N}_S = \{(\boldsymbol{t}_l^1, \boldsymbol{t}_r^1), \ldots, (\boldsymbol{t}_l^{(k_1)}, \boldsymbol{t}_r^{(k_1)})\}$ the set of negative examples for the synonymous word pairs. A similar set $\mathscr{N}_A = \{(\boldsymbol{t}_l^1, \boldsymbol{t}_r^1), \ldots, (\boldsymbol{t}_l^{(k_2)}, \boldsymbol{t}_r^{(k_2)})\}$ be defined for the antonymous pairs.
The negative counterpart of a word pair $(\boldsymbol{x}_l, \boldsymbol{x}_r)$ is selected from the other words in batch $\mathscr{B}$. For synonymous word pairs, $\boldsymbol{t}_l$ is the nearest neighbor of $\boldsymbol{x}_l$, while $\boldsymbol{t}_r$ is the nearest neighbor of $\boldsymbol{x}_r$. Similarly, for antonymous word pairs, $\boldsymbol{t}_l$ is selected to be the vector with the largest distance to $\boldsymbol{x}_l$, while $\boldsymbol{t}_r$ is the vector with the largest distance to $\boldsymbol{x}_r$. The cost function of the model is given by:

$$\mathscr{J}(\mathscr{B}_S) = \sum_{(\boldsymbol{x}_l, \boldsymbol{x}_r) \in \mathscr{B}_S} \left(\tau(\delta_{syn} + \boldsymbol{x}_l^{\mathsf{T}} \boldsymbol{t}_l - \boldsymbol{x}_l^{\mathsf{T}} \boldsymbol{x}_r) + \tau(\delta_{syn} + \boldsymbol{x}_r^{\mathsf{T}} \boldsymbol{t}_r - \boldsymbol{x}_l^{\mathsf{T}} \boldsymbol{x}_r)\right) \tag{4}$$

$$\mathscr{J}(\mathscr{B}_A) = \sum_{(\boldsymbol{x}_l, \boldsymbol{x}_r) \in \mathscr{B}_A} \left(\tau(\delta_{ant} + \boldsymbol{x}_l^{\mathsf{T}} \boldsymbol{x}_r - \boldsymbol{x}_l^{\mathsf{T}} \boldsymbol{t}_l) + \tau(\delta_{ant} + \boldsymbol{x}_l^{\mathsf{T}} \boldsymbol{x}_r - \boldsymbol{x}_r^{\mathsf{T}} \boldsymbol{t}_r)\right) \tag{5}$$

$$\mathscr{J}(\mathscr{B}) = \mathscr{J}(\mathscr{B}_S) + \mathscr{J}(\mathscr{B}_A) + \text{regularization term} \tag{6}$$

$\tau(x)$ denotes the **hinge loss** and δ_{syn} is a hyper-parameter specifying how much closer the synonymous words should be in contrast to the negative examples (δ_{ant} is defined analogously for antonymous word pairs). For intuition, consider equation 4 which shows the part of the cost function concerning synonymous constraints. If $\boldsymbol{x}_l$ and $\boldsymbol{x}_r$ are nearby vectors, their dot product will be large. This is desirable, since the two words are synonymous. Subtracting this quantity reduces the cost. At the same time, a large dot product of $\boldsymbol{x}_l$ with its negative example $\boldsymbol{t}_l$ is punished by adding the respective term. In their experiments the authors minimize the objective function using the **AdaGrad** optimization technique (Duchi et al., 2011). Training for five epochs without early stopping was found to be sufficient to obtain sensibly fine-tuned word representations. The authors could produce significant quality improvements for low-resource languages and observed enhanced performance in downstream tasks.

The word embeddings presented in this chapter play an important role, since they form the basis for sentence embedding algorithms which will be covered in chapter 4 on page 12. Table 1 summarizes how these word embeddings were obtained:

Nr.	Embedding	Embedding source					Download	Training details		
		EN	DE	RU	TR	KA		Dimen.	Window size	Architecture
❶	*word2vec*	⬇	⬇	⚙	⚙	⚙	Link1, Link2	300	10	CBOW
❷	*FastText*	⬇	⬇	⬇	⬇	⬇	Link	300	-	-
❸	*Attract-Repel*	⬇	⬇	⬇	⬇	⬇	Link	300	-	-

Table 1: Overview of word embedding acquisition. The ⬇ symbol indicates that the model was downloaded, whereas the ⚙ symbol refers to a trained model.

Availability of pre-trained word embeddings. Pre-trained embedding spaces for *FastText* and *Attract-Repel* are already publicly available for all target languages. These word vectors were thus downloaded from the respective sources (indicated by the ⬇ symbol in table 1). We found *word2vec* vectors for English and German, but unfortunately not for Georgian. This is why we trained these embeddings from scratch (indicated by the ⚙ symbol in table 1). For Turkish, only 200-dimensional vectors were available. Different embedding dimensionalities render the results less comparable as already pointed out by (Eger et al., 2019). Therefore, we chose to train Turkish embeddings as well. Although trained vector representations exist for Russian, we decided to proceed analogously, in order to have similar conditions among the low-resource languages.

Training details. We trained all *word2vec* embeddings on Wikipedia dumps for the respective languages (Russian, Turkish and Georgian). The dumps are provided by Wikimedia[10] in an Extensible Markup Language (XML) format. In order to get hold of the plain text which is needed for training, these XML files have to be parsed. A module for the Python programming language called `WikiExtractor`[11] was used for this purpose. Table 2 shows the sizes of the Wikipedia dumps as measured by the number of sentences.

Mikolov and colleagues provide the official C-implementation of *word2vec* on GitHub.[12] For training, we did not change the default hyper-parameters of the script. The training setting is as follows: First of all, the *CBOW* architecture was used and the window size was set to $m = 10$. We chose a dimensionality of the resulting vectors of $d = 300$. The training process took about 2 hours per language on a Graphics Processing Unit (GPU) computer, where the reduced amount of training time is due to smaller data sets as can be seen in table 2.

Language	# sentences	Version
English	97,285,870	20.02.2019
German	40,729,339	20.09.2018
Russian	25,324,234	01.02.2019
Turkish	3,797,955	01.01.2019
Georgian	1,047,780	20.07.2016

Table 2: Number of sentences contained in the Wikipedia dumps for each language. As can be seen, among the languages considered Georgian is by far the one with the fewest resources.

[10] `https://dumps.wikimedia.org/` (retrieved: September 20, 2019)
[11] `https://github.com/attardi/wikiextractor` (retrieved: September 20, 2019)
[12] `https://github.com/tmikolov/word2vec` (retrieved: September 20, 2019)

In this chapter the concept of word embeddings was introduced. Since most machine learning algorithms are not designed for natural language input, researchers began to devise techniques to obtain vectorial representations of language which are called language embeddings. Mikolov and colleagues introduced a seminal approach they called *word2vec*. It can be trained efficiently on large corpora and captures not only syntactic but also semantic aspects of words. The fact that simple algebraic operations on these vectors suffice to answer similarity questions popularized this algorithm in the NLP community.

To account for some drawbacks of the *word2vec* algorithm, other techniques and extensions were proposed. *FastText* extends the *Skip-Gram* model to character *n*-grams which enables it to cope with OOV-words in a more graceful fashion. *Attract-Repel* explicitly includes linguistic constraints based on synonymy and antonymy relationships to fine-tune existing word embeddings. The idea behind this approach is that synonymous words should be mapped close to each other in the embedding space while antonymous words should be as far apart as possible.

Finally, an overview provided information about where and how the word embeddings were obtained. Pre-trained models for *FastText* and *Attract-Repel* embeddings were available for all target languages and were therefore downloaded from the official websites. Suitable *word2vec* vectors were available only for English and German. For the low-resource languages Russian, Turkish and Georgian they were trained from scratch using the official implementation provided by Mikolov and colleagues.

4 Sentence Embedding Algorithms

4.1 Introduction

Due to the significant impact word embeddings had on the performance of machine learning techniques in the context of NLP, the question arose whether it is possible to encode even larger spans of text in terms of a single vector. (Le and Mikolov, 2014) were among the first researchers to introduce embeddings for sentences, paragraphs and even entire documents. This chapter gives an introduction to common sentence embedding algorithms. (Yang et al., 2018) distinguish between two major approaches: They differentiate between **non-parameterized** methods which combine underlying word embeddings and **parameterized** methods which train sentence embeddings from scratch. Table 3 gives an overview of common algorithms (*not exhaustive*):

Non-parameterized methods		Parameterized methods	
Algorithm	**Source**	**Algorithm**	**Source**
❶ *Vanilla average*	*- no author found -*	❶ *Skip-Thought*	(Kiros et al., 2015)
❷ *Concatenated p-means*	(Rücklé et al., 2018)	❷ *Quick-Thought*	(Logeswaran and Lee, 2018)
❸ *GEM*	(Yang et al., 2018)	❸ *sent2vec*	(Pagliardini et al., 2018)
❹ *SIF*	(Arora et al., 2017)	❹ *InferSent*	(Conneau et al., 2017)
		❺ *SBERT*	(Reimers and Gurevych, 2019)
		❻ *LASER*	(Artetxe and Schwenk, 2018)

Table 3: Types and examples of sentence embedding algorithms. Non-parameterized methods combine existing word embeddings to obtain a sentence representation whereas parameterized methods train sentence embeddings from scratch.

The rest of this chapter is organized as follows: Section 4.2 introduces some examples of non-parameterized algorithms which are going to be used in the experiments. Section 4.3 on the following page then presents their parameterized counterparts. Analogously to the previous chapter, details are given where and how the individual sentence encoders were obtained (cf. section 4.4 on page 18 for more information).

4.2 Non-parameterized Sentence Embeddings / Compositional Models

4.2.1 Vanilla Average

The most straightforward approach to generate a sentence representation is the *vanilla average* technique. The embeddings of the words in the sentence to be encoded are averaged using a simple **arithmetic mean**: $\boldsymbol{v}_s = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{v}_{w_t}$, where T denotes the length of the sentence. While this approach is very intuitive and easy to implement, it has severe shortcomings: ❶ It **neglects the word order**, therefore it is also referred to as a Bag of Words (BOW) approach. ❷ Furthermore, **stop-words and more informative words are weighted equally** (Le and Mikolov, 2014). Figure 5 shows an exemplary sentence:

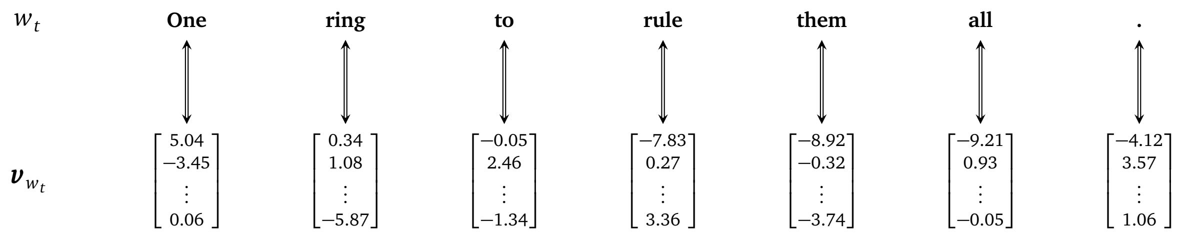

w_t
One
ring
to
rule
them
all
.
$\boldsymbol{v}_{w_t}$
$\begin{bmatrix}5.04\\-3.45\\\vdots\\0.06\end{bmatrix}$
$\begin{bmatrix}0.34\\1.08\\\vdots\\-5.87\end{bmatrix}$
$\begin{bmatrix}-0.05\\2.46\\\vdots\\-1.34\end{bmatrix}$
$\begin{bmatrix}-7.83\\0.27\\\vdots\\3.36\end{bmatrix}$
$\begin{bmatrix}-8.92\\-0.32\\\vdots\\-3.74\end{bmatrix}$
$\begin{bmatrix}-9.21\\0.93\\\vdots\\-0.05\end{bmatrix}$
$\begin{bmatrix}-4.12\\3.57\\\vdots\\1.06\end{bmatrix}$

Figure 5: An example of a sentence of length seven ($T = 7$). The single word vectors are averaged in order to obtain a sentence embedding.

Vanilla average sentence embeddings are commonly used as a baseline. More sophisticated algorithms should be able to outperform this naïve method. In fact, this algorithm is an instantiation of a more general family of algorithms which (Shen et al., 2018) call SWEMs. An arithmetic mean is not the only pooling mechanism which can be applied. Other strategies include *min-pooling*, *max-pooling* or *hierarchical pooling*. The latter technique makes use of a sliding window which is shifted over the input sentence, where all word representations in the window are pooled until finally, the resulting pooled vectors for each window are in turn combined to form the ultimate sentence vector.

4.2.2 Concatenated Power Means

(Rücklé et al., 2018) propose to generalize the idea of average sentence embeddings to a concatenation of several power-means. The concatenation aims to capture more information about the sentence structure and is an attempt to make the comparison to trained models fairer. As was found by the authors, more complex and high-dimensional models like *InferSent* are often directly compared to lower-dimensional average embeddings. Equation 7 depicts the general form of a power mean. The equation introduces a hyper-parameter p which specifies the type of average to be computed. As can directly be seen, the arithmetic mean is obtained by setting $p = 1$. Different values for p yield different means. E. g. if $p = -1$, the output of the formula is the **harmonic mean**. For p approaching (negative) ∞ the result is going to be the **maximum** (**minimum**) of the input data. The following paragraphs describe how this idea can be used to generate sentence embeddings which are more expressive.

$$\overline{x} = \left[\frac{x_1^p + x_2^p + \cdots + x_T^p}{T} \right]^{1/p} \tag{7}$$

Let $\boldsymbol{S}^{(j)} = [\boldsymbol{v}_{w_1}^{(j)}, \boldsymbol{v}_{w_2}^{(j)}, \ldots, \boldsymbol{v}_{w_T}^{(j)}] \in \mathbb{R}^{d_j \times T}$ be the embedding matrix of a sentence $s = \langle w_1, w_2, \ldots, w_T \rangle$ of length T. j is the index variable into the set of l embedding spaces $\mathbb{E} = \{\mathbb{E}^{(j)}\}_{j=1}^{l}$ and d_j denotes the dimensionality of the word embeddings in the respective embedding spaces. $\mathbb{E}^{(j)}$ contains all word vectors that were trained by the same method. For example all *word2vec* vectors constitute one embedding space whereas all *FastText* vectors lie in a different embedding space. Further, let $f_p(\boldsymbol{S}^{(j)})$ with $f_p : \mathbb{R}^{d_j \times T} \mapsto \mathbb{R}^{d_j}$ be a mapping of a sentence embedding matrix to a single p-mean vector. The sentence representation for one embedding space is computed by applying function f_p k times with different values for p and successively concatenating the results (concatenation is denoted by the $\oplus$ symbol): $\boldsymbol{v}_s^{(j)} = f_{p_1}\left(\boldsymbol{S}^{(j)}\right) \oplus f_{p_2}\left(\boldsymbol{S}^{(j)}\right) \oplus \cdots \oplus f_{p_k}\left(\boldsymbol{S}^{(j)}\right)$. The ultimate sentence embedding $\boldsymbol{v}_s$ is generated by concatenating the representations from all embedding spaces $\mathbb{E}^{(j)}$.

4.3 Parameterized Sentence Embeddings / Trained Encoders

4.3.1 Skip-Thought and Quick-Thought

Skip-Thought. Inspired by the *Skip-Gram* model, (Kiros et al., 2015) proposed a sentence embedding algorithm named *Skip-Thought*. Given a sentence s_i, the model's objective in the training phase is to predict the two neighboring sentences s_{i-1} and s_{i+1}, respectively. Thus, the input to the model consists of a tuple of three sentences which have to be taken from an **ordered corpus**, i. e. nearby sentences have to be related. For example consider the following three sentences:

$\langle$ **I got back home** $\rangle_{s_{i-1}}$, $\langle$ **I could see the cat on the steps** $\rangle_{s_i}$ and $\langle$ **This was very strange** $\rangle_{s_{i+1}}$

Especially well suited are book corpora which provide large amounts of contiguous text. (Kiros et al., 2015) trained the model on the *BookCorpus* data set by (Zhu et al., 2015). The training corpus is not annotated, hence *Skip-Thought* belongs to the family of unsupervised machine learning algorithms. Figure 6 on the following page depicts the model architecture: The middle sentence s_i (*'I could see the cat on the steps'*) is first encoded into a compact representation $\boldsymbol{h}_i$ which is then passed on to two separate decoders which predict the surrounding sentences s_{i-1} (*'I got back home'*) and s_{i+1} (*'This was very strange'*). At each decoding step, the model is fed with its output one time step in the past as well as the vector representation of the middle sentence $\boldsymbol{h}_i$ (indicated by the gray dashed arrows in figure 6 on the next page). Sometimes the model output is replaced by the gold label word. This method is called **teacher forcing** (Williams

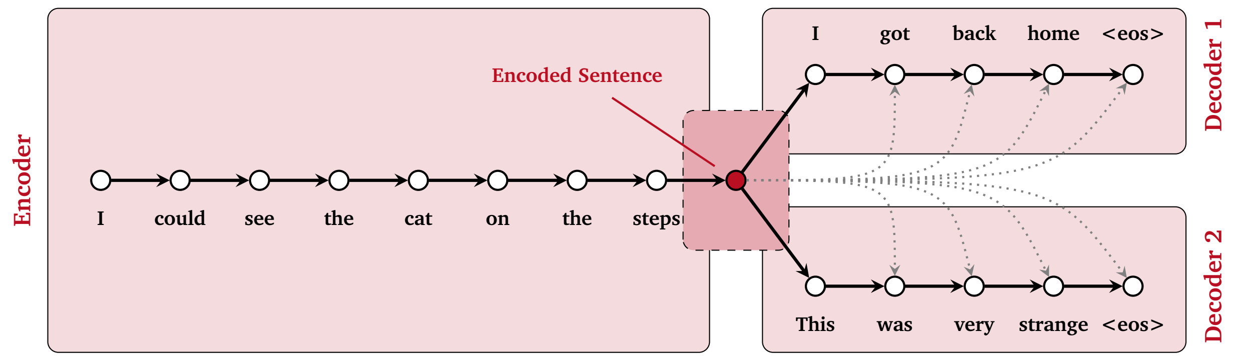


Figure 6: The *Skip-Thought* model encodes the middle sentence 'I could see the cat on the steps' and predicts the surrounding two sentences. The figure was taken and adapted from (Kiros et al., 2015).

and Zipser, 1998). During the training process, the model aims to minimize the reconstruction error by maximizing the log-probability of the next word to be decoded, given all previously decoded words as well as the center sentence representation $\boldsymbol{h}_i$. This error is back-propagated not only through the decoders but also through the encoder component, thus forcing the encoder to put as much information about the center sentence s_i as possible into $\boldsymbol{h}_i$. After training, the fixed-length encoder output represents the desired sentence embedding. Only the encoder is needed to embed new sentences, the decoders can be discarded. Their only purpose was to provide the training signal needed in the training procedure. (Kiros et al., 2015) use Gated Recurrent Units (GRUs) (Chung et al., 2014) for both, encoder and decoders.

Quick-Thought. Based on the *Skip-Thought* approach, (Logeswaran and Lee, 2018) proposed a different method called *Quick-Thought*. One of the major problems with the earlier invented *Skip-Thought* model is its word level objective which requires an excessive amount of computation. This entails a considerable slow-down of the algorithm and as a consequence, the training data sets and the vocabulary sizes have to be kept low which inhibits effective learning. *Quick-Thought* provides a remedy by using a sentence-level objective which eschews much of the computational cost and can therefore be trained faster and on larger training data sets.[13] For their model, (Logeswaran and Lee, 2018) rephrase the encoder-decoder approach as a **multi-class classification problem**: Given a sentence, the *Quick-Thought* algorithm distinguishes a context sentence among a set of other contrastive sentences by only taking the respective sentence embeddings into account (cf. figure 7).

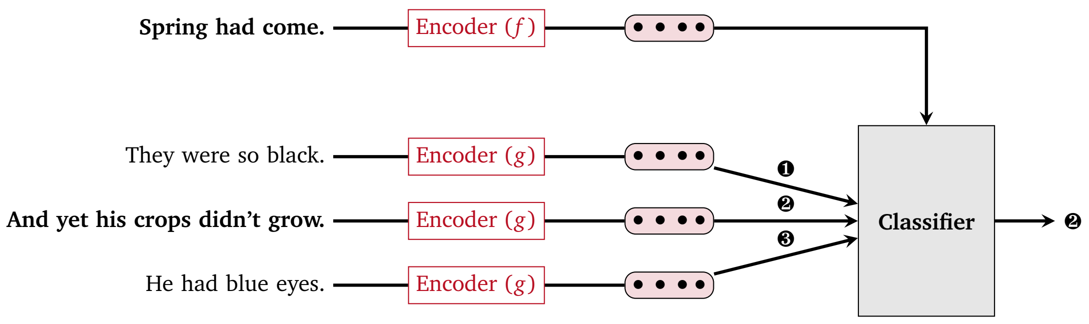


Figure 7: The *Quick-Thought* model reformulates the encoder-decoder approach as a multi-class classification problem. Given an input sentence and a set of candidate sentences (the set comprises one target sentence and other contrastive sentences) a classifier learns to predict the correct context sentence. The figure was taken and adapted from (Logeswaran and Lee, 2018).

Another issue of the word-level objective is that **sentences can have a similar semantic meaning and yet exhibit a completely different surface form**. Already (von Humboldt, 1836) observed that natural language makes *'infinite use of finite means'*. Next to the semantic meaning of a sentence, standard encoder-decoder models also encode the surface form of the sentence which is usually not desired. The embeddings should be insusceptible to such superficial aspects. *Quick-Thought* avoids this due to its training objective. (Logeswaran and Lee, 2018) also trained their models on the *BookCorpus* data set (Zhu et al., 2015) and additionally on the *UMBC* corpus (Han et al., 2013), which consists of texts taken from crawled web-pages.

[13] Hence the name ***Quick**-Thought*.

4.3.2 InferSent

The *InferSent* algorithm proposed by (Conneau et al., 2017) differs from the previous methods in that it uses a supervised learning approach. The intention of *InferSent* is to provide universal sentence embeddings, **i. e. they can be employed in a task-agnostic manner.**[14] The authors address two major questions in their paper: Since the algorithm needs labeled training data, it is first of all important to ask which training data is suitable for learning high-quality sentence embeddings. The second, but not less important question is about the architecture which should be used to facilitate the learning process in an optimal way. Concerning the first question, (Conneau et al., 2017) claim that **the NLI task is adequate for the training of universal sentence embeddings**, since good performance in this task requires that the sentence representations incorporate semantic relationships to a large extent. The authors decided to use the *SNLI* data set (+ *MultiNLI*) collected by (Bowman et al., 2015): Given two sentences, s_1 and s_2, NLI is the task of deciding whether s_2 entails s_1, contradicts s_1 or whether there is no connection at all. Usually, s_1 is referred to as the **premise** sentence and s_2 as the **hypothesis** sentence. Table 4 shows some exemplary sentence pairs with their corresponding annotations. The sentence pairs were labeled by five human annotators. Their judgments are listed in the last column of of table 4, where 'E' stands for entailment, 'C' for contradiction and 'N' for neutral. As can be seen, even human annotators face problems in achieving a unanimous vote. The gold labels were therefore obtained by a majority vote.

Nr	**Premise** (sentence s_1)	**Hypothesis** (sentence s_2)	Label	Judgments
❶	A man inspects the uniform of a figure in some East Asian country.	The man is sleeping.	`Contradiction`	`C C C C C`
❷	An older and younger man smiling.	The men are smiling and laughing at the cats playing on the floor.	`Neutral`	`N N E N N`
❸	A black race car starts up in front of a crowd of people.	A man is driving down a lonely road.	`Contradiction`	`C C C C C`
❹	A soccer game with multiple males playing.	Some men are playing sport.	`Entailment`	`E E E E E`
❺	A smiling costumed woman is holding an umbrella.	A happy woman in a fairy costume holds an umbrella.	`Neutral`	`N N E C N`

Table 4: Example sentence pairs taken from the SNLI data set. The 'judgments' column shows the votes among the five annotators whereas the 'label' column contains the gold label (majority vote). The example was taken and adapted from `https://nlp.stanford.edu/projects/snli/` (retrieved: September 04, 2019).

The overall neural architecture underlying the *InferSent* model is depicted in figure 8 on the next page (left). It consists of two parts: ❶ An encoder whose task is to encode a given sentence into a compact representation and ❷ a classifier which gets a pair of two encoded sentences $(\boldsymbol{p}, \boldsymbol{h})$ and two additional features calculated on top of these two representations (absolute difference of $\boldsymbol{p}$ and $\boldsymbol{h}$: $|\boldsymbol{p}-\boldsymbol{h}|$ and the Hadamard/element-wise product of the two embedding vectors involved: $\boldsymbol{p} \odot \boldsymbol{h}$) as input and subsequently outputs one of the three possible classes: `Entailment`, `Contradiction` or `Neutral`. The classifier which operates on top of the encoder is a standard fully connected Multi-Layer Perceptron (MLP). For the encoder, (Conneau et al., 2017) consider seven possible architectures: E. g. **Bidirectional Long Short-Term Memories (BiLSTMs) with max/mean-pooling** (cf. figure 8 on the following page, right), **hierarchical Convolutional Neural Networks (CNNs)** or **self-attentive networks**. The evaluation results indicate the best performance for the BiLSTM (max-pooling) architecture. Therefore, this type of encoder is going to be considered in the following.

Long Short-Term Memories (LSTMs) were originally proposed by (Hochreiter and Schmidhuber, 1997). They belong to the family of RNNs and were especially designed to tackle the **vanishing gradient problem** (Hochreiter, 1998) by using a more complicated recurrent unit than the one used in standard RNNs. Given an input sequence $\{\boldsymbol{x}_t\}_{t=1}^{T}$ of length T, an LSTM generates a series of hidden states $\boldsymbol{h}_t$. At each time step t, it takes the current input $\boldsymbol{x}_t$ as well as the previous hidden state $\boldsymbol{h}_{t-1}$ into account. *InferSent* uses a concatenation of two LSTMs operating in opposite directions. Models with this architecture are called BiLSTMs. The input to the encoder is given by word embeddings. The original version

[14] Many word or sentence embeddings are trained for specific applications and perform poorly if they are transferred to tasks other than the one they were designed for. The term 'task-agnostic' means that the embeddings are trained without specific downstream applications in mind which makes them usable in a wide range of tasks.

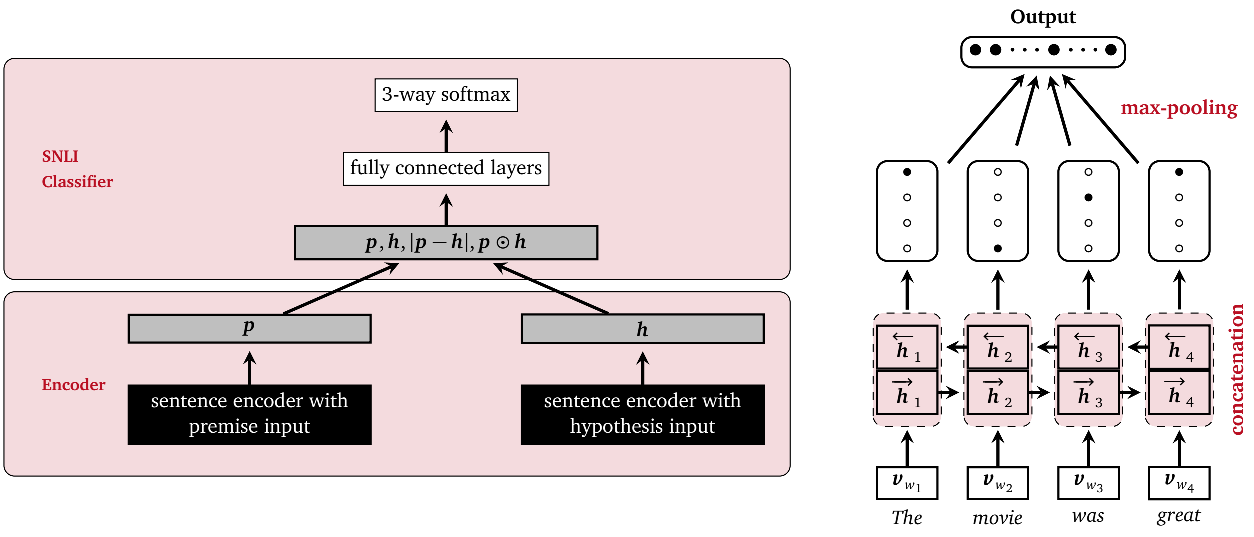


Figure 8: Left: Global *InferSent* architecture. Right: *InferSent* encoder architecture. The images were taken and adapted from (Conneau et al., 2017).

InferSent1 uses *GloVe* embeddings (Pennington et al., 2014), whereas *InferSent2* makes use of *FastText* word embeddings (Bojanowski et al., 2017).

4.3.3 BERT

(Devlin et al., 2018) propose a model called BERT which is based on the transformer architecture introduced by (Vaswani et al., 2017). The model consists of a stack of such transformers (cf. figure 9). The model training consists of two phases:

Phase ❶: In the first step, the embeddings are pre-trained using a **Masked Language Model (MLM)** which is inspired by the **Cloze task** (Taylor, 1953): Approximately 15 % of the input words are randomly masked which subsequently have to be predicted by the model. Masking allows BERT to incorporate left as well as right contexts. This architectural characteristic distinguishes it from the *OpenAI GPT* model published by (Radford et al., 2018) which only uses the left-context. A second task – called **'two sentences task'** – is used to pre-train the representations. This task is similar to the NLI task where the classifier has to predict whether or not the second sentence is a follow-up sentence of the first one.

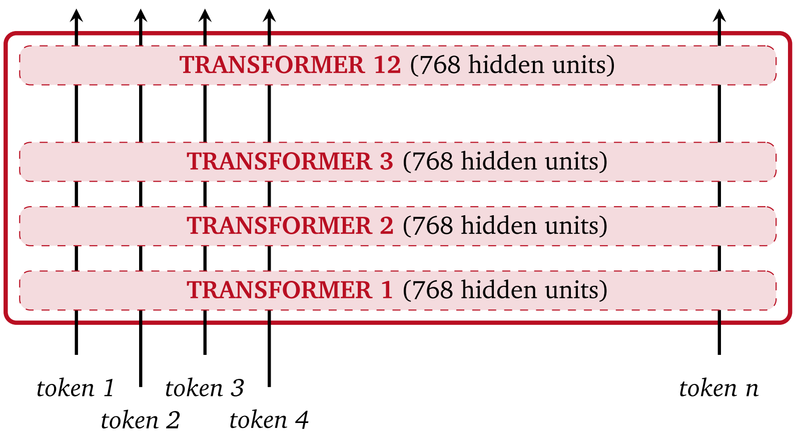

TRANSFORMER 12 (768 hidden units)
TRANSFORMER 3 (768 hidden units)
TRANSFORMER 2 (768 hidden units)
TRANSFORMER 1 (768 hidden units)
token 1
token 2
token 3
token 4
token n

Figure 9: The $BERT_{BASE}$ encoder stack using transformers as a basis. The figure was taken and adapted from `http://jalammar.github.io/illustrated-bert/` (retrieved: September 04, 2019).

Phase ❷: In an optional second step, the pre-trained embeddings can further be **fine-tuned on the downstream task for which the embeddings are used**. The authors report results for two different architectures, $BERT_{BASE}$ and $BERT_{LARGE}$. The architectural characteristics of these two models are as follows:

Model	# transformers	# hidden units	# attention heads
$BERT_{BASE}$	12	768	12
$BERT_{LARGE}$	24	1,024	16

The model is primarily used to create **contextualized word embeddings** by using the transformer activations across different layers for one word. We obtain sentence representations from the model by using the activations of the last transformer layer (alternative: an average of several transformer layers). In the context of this thesis a multi-lingual version of the model is used which was trained for 104 different languages and it is publicly available on GitHub (without fine-tuning as described in step ❷).[15] Recently, (Reimers and Gurevych, 2019) have introduced a new *BERT* version they call *SBERT*. This model was designed specifically for the purpose of creating sentence embeddings. Unfortunately, time constraints prevented us from using this technique (due to its novelty).

4.3.4 LASER / Massively Multilingual Sentence Embeddings

Massively multilingual sentence embeddings are a rather new approach introduced by (Artetxe and Schwenk, 2018). The algorithm is implemented in the *LASER* library which is why the representations are referred to as *LASER* embeddings throughout this thesis. As the name already implies, this approach creates a **single shared embedding space for all the languages considered** during the training procedure. Therefore, only one model is needed to embed sentences in various languages.

Figure 10 shows the architecture of the model which is similar to (Schwenk, 2018). It leverages an **encoder-decoder approach** where the encoder is given by a BiLSTM. The words of the input sentences are represented using a joint Byte Pair Encoding (BPE) vocabulary which was trained on one big corpus obtained by concatenating all training resources for all languages (aligned parallel corpora).

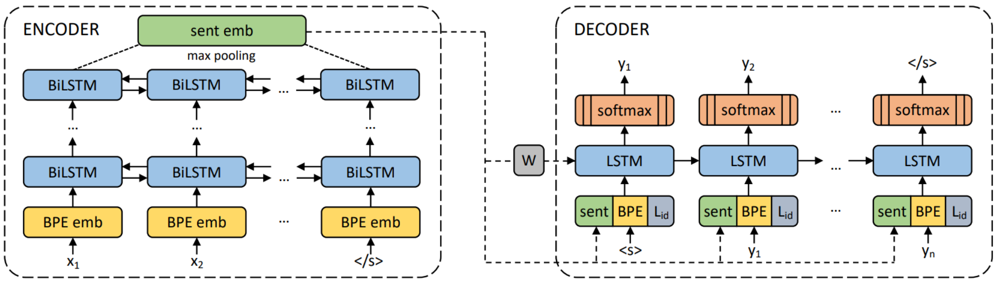


Figure 10: The architecture of the 'massively multilingual sentence embeddings' which are part of the *LASER* library. The image was taken from (Artetxe and Schwenk, 2018).

The sentence representation is then passed on to the decoder which generates a sequence of words similar to the *Skip-Thought* approach. However, this model additionally gets a language flag as input indicating which language the decoder has to produce. With their approach the authors attempt to create **'universal language-agnostic sentence representations'**, i. e. the embeddings are designed to work well across many tasks and across multiple languages.

4.3.5 Random Sentence Encoders

Recent research conducted by (Wieting and Kiela, 2019) revealed that **random sentence representations provide a solid baseline** for more sophisticated algorithms. The authors investigate the results of several models whose parameters are chosen randomly without any training. It was found that while there is a gap between the performance of such random encoders and the representations generated by trained models, it is much smaller than one would expect. Three architectures are proposed in the paper: ❶ *Bag of random Embedding Projections (BOREP)*, ❷ *random (Bi-)LSTMs* and ❸ *Echo State Networks (ESNs)* (Jaeger, 2001) out of which the first two are used in this thesis.

[15] `https://github.com/google-research/bert/blob/master/multilingual.md` (retrieved: September 04, 2019)

Analogously to the previous chapter, it is useful to summarize the training details for all the sentence embeddings considered in the context of the experiments. Table 5 on the next page gives an overview.

Non-parameterized methods. Recall, that non-parameterized methods do not require any training. Instead, they combine existing word embeddings (compositional models). The experiments will be conducted using all word embeddings discussed in section 3 on page 7 (*word2vec*, *FastText* and *Attract-Repel*). All of these word embedding spaces have a dimensionality of 300 which entails that the sentence embeddings produced by the compositional models have the same dimensionality. An exception is the *power means* embedding which is a concatenation of various means, where $p \in \{-\infty, 0, 1, 2, +\infty\}$. Furthermore, the code for the *hierarchical pooling* algorithm could not be found in the official repository, which is why it was implemented from scratch according to the paper. In the paper, a window size of $m = 5$ is used. Unfortunately, this information was discovered relatively late which is why the experiments were conducted with a different windows size of $m = 3$.

Parameterized methods. In the context of this thesis five trained parameterized methods are going to be evaluated: *InferSent*, *Quick-Thought*, *sent2vec*, *BERT* and *LASER*. Furthermore, there are two types of random encoders: *BOREP* and the *random BiLSTM*. These random methods have parameters (parameterized method), but they are not trained after a random initialization. These models will (next to the majority class) serve as a baseline.

Quick-Thought. The *Quick-Thought* models were trained on the Wikipedia dumps for the respective languages. The English model was also trained on Wikipedia and not downloaded from the official GitHub repository.[16] The reason for this is that there were some problems when trying to get the pre-trained model to work. Several issues regarding this problem have already been filed.[17] Furthermore, for the Turkish and Georgian models, the number of epochs trained was increased, since there is considerably less data available for these languages.

InferSent. We trained the *InferSent* models on the SNLI data set. Unfortunately, this data set is originally only available for the English language. Therefore, the resources had to be translated. This was achieved using the Google translation Application Programming Interface (API). A major problem in this context was that the number of translation requests is limited. Only a certain number of requests per day are allowed. If this limit is exceeded, the server responds with `Error Code 500: Too many Requests` and one is blocked for 24 h. In order to increase the speed, several sentences are concatenated which are jointly translated as a batch. Unfortunately, this entails that the translated sentences have to be mapped back to the original sentences (since the result returned is only one long string). It is crucial to assure that the number of sentences returned by the server is equal to the number of sentences fed into the translation procedure. If this is not the case, one risks that sentences and labels do not match anymore which would greatly impair the learning process.

In order to avoid this, the following strategy was employed: A special split token (a series of numbers) was introduced before concatenating the individual sentences in the original language. It was checked that this split token does not influence the quality of the translations in any harmful way. Further, the token was chosen such that it is not altered during the translation procedure (at least in the majority of the cases). When a translated batch arrives it is first split using this split token. The routine subsequently checks if this results in the expected number of sentences. If yes, the sentences are returned. If not, the split token is removed and the next attempt is to split at punctuation symbols (., !, ?). If this again fails, the sentences in that batch are translated one by one. The translations had to be done for Russian, Turkish and Georgian. A German translation of the SNLI data set was already available.[18]

Since the translation procedure nonetheless took a very long time, it was not possible to translate all of the approximately 550,000 sentence pairs for each language, which is why the training set for the low-resource languages was restricted to 60,000 sentence pairs. All models employed use the *InferSent2* version (using *FastText* word embeddings) and the encoder architecture leveraged was the BiLSTM with max-pooling, since this architecture achieved the best results in the experiments conducted by (Conneau et al., 2017). Training the models takes a very long time. The time for training the German model was about 4 days. The English model was again available for download.[19]

[16] https://github.com/lajanugen/S2V (retrieved: September 04, 2019)

[17] Cf. for example https://github.com/lajanugen/S2V/issues/10 or https://github.com/lajanugen/S2V/issues/8 (retrieved: September 17, 2019)

[18] https://public.ukp.informatik.tu-darmstadt.de/arxiv2018-xling-sentence-embeddings/translated-snli/en-de-translated-snli-4x.zip (retrieved: September 04, 2019)

[19] https://github.com/facebookresearch/InferSent (retrieved: September 04, 2019)

Nr.	Embedding	Lng	Source	Word emb	Training details				
					dim	# instances	batch size	# epochs	other params / comments
Non-parameterized methods									
①	*Vanilla average*	🇬🇪🇹🇷🇷🇺🇩🇪🇬🇧	▼	*all*	300	-	-	-	no training required
②	*p-means*	🇬🇪🇹🇷🇷🇺🇩🇪🇬🇧	▼	*all*	1,500	-	-	-	no training required, $p \in \{-\infty, 0, 1, 2, +\infty\}$
③	*SIF*	🇬🇪🇹🇷🇷🇺🇩🇪🇬🇧	▼	*all*	300	-	-	-	no training required
④	*GEM*	🇬🇪🇹🇷🇷🇺🇩🇪🇬🇧	▼	*all*	300	-	-	-	no training required
⑤	*hier. pooling*	🇬🇪🇹🇷🇷🇺🇩🇪🇬🇧	▼	*all*	300	-	-	-	no training required, window size $m = 3$
Parameterized methods									
❶	*Quick-Thought*	🇬🇧	⚙	*word2vec*	2,400	97,285,870	2,000	5	-
❷	*Quick-Thought*	🇩🇪	⚙	*word2vec*	2,400	40,729,339	1,000	5	-
❸	*Quick-Thought*	🇷🇺	⚙	*word2vec*	2,400	25,324,234	1,000	5	-
❹	*Quick-Thought*	🇹🇷	⚙	*word2vec*	2,400	3,797,955	100	5	-
❺	*Quick-Thought*	🇬🇪	⚙	*word2vec*	2,400	1,047,780	100	12	
❻	*InferSent*	🇬🇧	⭳	*FastText*	4,096	-	-	-	Version: *infersent2* , max-pooling, model: Link
❼	*InferSent*	🇩🇪	⚙	*FastText*	4,096	547,743	64	10	Version: *infersent2* , max-pooling
❽	*InferSent*	🇷🇺	⚙	*FastText*	4,096	60,000	64	15	Version: *infersent2* , max-pooling
❾	*InferSent*	🇹🇷	⚙	*FastText*	4,096	60,000	64	15	Version: *infersent2* , max-pooling
❿	*InferSent*	🇬🇪	⚙	*FastText*	4,096	60,000	64	15	Version: *infersent2* , max-pooling
⓫	*sent2vec*	🇬🇧	⭳	*FastText*	700	-	-	-	model: `sent2vec_wiki_bigrams`, Link
⓬	*sent2vec*	🇩🇪	⚙	*FastText*	700	40,729,339	-	9	`lr: 0.2, wordNgrams: 2, neg: 10, dropoutK: 4`
⓭	*sent2vec*	🇷🇺	⚙	*FastText*	700	25,324,234	-	9	`lr: 0.2, wordNgrams: 2, neg: 10, dropoutK: 4`
⓮	*sent2vec*	🇹🇷	⚙	*FastText*	700	3,797,955	-	9	`lr: 0.2, wordNgrams: 2, neg: 10, dropoutK: 4`
⓯	*sent2vec*	🇬🇪	⚙	*FastText*	700	1,047,780	-	9	`lr: 0.2, wordNgrams: 2, neg: 10, dropoutK: 4`
⓰	*BERT*	🇬🇪🇹🇷🇷🇺🇩🇪🇬🇧	⭳	-	768	-	-	-	multilingual model: Link
⓱	*LASER*	🇬🇪🇹🇷🇷🇺🇩🇪🇬🇧	⭳	-	1,024	-	-	-	multilingual model: Link
⓲	*RandSent BOREP*	🇬🇪🇹🇷🇷🇺🇩🇪🇬🇧	🪄	*FastText*	4,096	-	-	-	-
⓳	*RandSent BiLSTM*	🇬🇪🇹🇷🇷🇺🇩🇪🇬🇧	🪄	*FastText*	8,192	-	-	-	-

Table 5: Overview of sentence embedding acquisition separated by non-parametric and parametric approaches. Legend: ▼ Compositional methods, ⭳ downloaded models, ⚙ trained models, 🪄 random encoders

sent2vec. The models for the *sent2vec* approach were trained on Wikipedia (German, Russian, Turkish and Georgian). The values for the hyper-parameters can be read off from table 5 on the previous page. The training procedure was rather fast. The English model was downloaded using a link from GitHub.[20]

BERT. Training *BERT* is only possible if one has a Tensor Processing Unit (TPU) at one's disposal. This was not the case. But a multilingual model is offered on GitHub.[21] The model used is called `BERT-Base, Multilingual Cased`. It is applicable to 104 languages and has about 110 million parameters.

LASER. The model is multilingual and available in pre-trained form. It was downloaded and installed following the instructions on GitHub.[22]

4.5 Summary

This chapter introduced several sentence embedding algorithms. Generally speaking, there is a distinction between non-parameterized and parameterized sentence embeddings. The first kind of algorithms makes use of word embeddings as a basis and perform some kind of pooling operation in order to obtain a sentence embedding. The more sophisticated parameterized models which are often realized using neural architectures train sentence embeddings from scratch either in an unsupervised or supervised fashion.

The final section then gave insights into the process of sentence embedding acquisition. Models for the English language were mostly found to be publicly accessible in a pre-trained manner and were therefore downloaded from the official websites. Especially models for the low-resource languages Russian, Turkish and Georgian were not available which is why we had to train them from scratch (*sent2vec*, *Quick-Thought* and *InferSent*).

[20] `https://github.com/epfml/sent2vec` (retrieved: September 04, 2019)
[21] `https://github.com/google-research/bert/blob/master/multilingual.md` (retrieved: September 04, 2019)
[22] `https://github.com/facebookresearch/LASER` (retrieved: September 04, 2019)

5 Probing Tasks

5.1 Introduction

Sentence embeddings have become ubiquitous. Despite the abundance of embedding algorithms, several authors (Adi et al., 2017; Veldhoen et al., 2016; Zhu et al., 2018, inter alia) noticed that the knowledge about which linguistic features are learned by these embeddings is fairly limited. To tackle this problem, they introduced the first simple evaluation tasks. Later, (Conneau et al., 2018a) made use of these ideas to systematically test sentence representations. The authors were the first to introduce the concept of probing tasks at a larger scale. This chapter provides details about the tasks used in this thesis.

This chapter is organized as follows: Section 5.2 introduces the probing tasks already used in the literature which are categorized into three different types: Surface tasks, syntactic tasks and semantic tasks. To allow for a sensible assessment of the applicability of the probing tasks in the target languages considered in this thesis (cf. section 5.4 on page 27), it is crucial to understand the grammatical characteristics of each language, which is why the target languages are briefly outlined in section 5.3 on page 24. Finally, section 5.5 on page 31 supplies information about the probing task selection and further shows how the respective tasks were designed and how training data sets were acquired.

5.2 Presentation of Probing Tasks

Probing tasks have become a common tool in the evaluation of sentence embeddings. According to (Conneau et al., 2018a), **a probing task is defined to be a *'classification problem that focuses on simple linguistic properties of sentences'***. Here, the term 'linguistic concepts' comprises all interesting properties of sentences which define their structure or meaning. Generally speaking, the evaluation using probing tasks works as follows:

Given an embedding mechanism and a data set which is labeled with the linguistic property of interest, each sentence is first encoded into a vector of fixed length. The resulting vector representations are subsequently fed into a simple classifier, e. g. an MLP or a logistic regression model. As usual, the error gradients are back-propagated through the network, but **the weights of the encoding model are frozen and therefore not adjusted during the procedure**. The ultimate test score (e. g. accuracy or any other evaluation metric) of the model trained on top of the embeddings can then be considered and indicator of the extent to which the property is encoded in the representations. If the score is high, information about the linguistic property must either implicitly or explicitly be encoded in the vector representations. If the embeddings lack this information, one has to expect a performance score around the chance level (or majority class). Similarly to (Conneau et al., 2018a), all tasks are in the following categorized into **surface tasks** (cf. section 5.2.1), **syntactic tasks** (cf. section 5.2.2 on the next page) and **semantic tasks** (cf. section 5.2.3 on page 23). **However, the classification for some tasks can be ambiguous, since they cover syntactic as well as semantic aspects of sentences.**

5.2.1 Probing Tasks for superficial Information

Surface information tasks probe sentence embeddings whether or not they retain information which can be read off from the sentences themselves, **without requiring any additional knowledge about the language** (e. g. grammar, vocabulary, etc.). A review of the literature revealed the following surface probing tasks:

❶ **Sentence length (SentLen).** This task was introduced by (Adi et al., 2017), its goal is to predict the length of a sentence given its vector representation. Instead of predicting the exact length, the authors introduced a set of bins to simplify the classification task. The following bins were used: $[0;4]$, $[5;8]$, $[9;12]$, $[13;16]$, $[17;20]$, $[21;25]$, $[26;29]$, $[30;33]$, $[34;70]$, $[71;\infty[$. (Conneau et al., 2018a) adopted this task without modifications. A variant of this task is the LenMismatch (length mismatch) task introduced by (Naik et al., 2018). The input to their model consists of two sentence representations and the classifier is tasked with predicting whether the two sentences have equal length or not. This gives a binary classification problem.

❷ **Word content (WC).** (Adi et al., 2017) were the first to propose this surface probing task. Embeddings are tested if they encode information about individual words in the sentence. The classifier is fed with the sentence representation as well as a word embedding. It has to give a binary answer whether or not the word represented by the word embedding is contained in the sentence. However, this task formulation violates the criteria for probing tasks postulated by (Conneau et al., 2018a) (*'no additional input to the classifiers'*) which is why they rephrased the task: They sampled 1k mid-frequency words from the training corpus and created a data set, where each sentence contains exactly one word from the set of 1k words previously sampled. The task is to predict which one the sentence contains.

❸ **Word overlap (WOverlap).** Introduced by (Naik et al., 2018). Given two sentence embeddings, a classifier has to decide whether the sentences have a considerable amount of words in common or not. The authors designed the task as a distraction test, since sentence pairs with little overlap are more likely to be erroneously labeled as `NEUTRAL` in the NLI task. Again, this task does not comply to Conneau's cirteria.

5.2.2 Probing Tasks for syntactic Information

The following probing tasks involve the syntax of sentences, i. e. **the way how smaller linguistic entities like words or syllables are arranged in order to form more complex constructs like sentences**. Syntax includes for example the order of words which is strict in some languages and more flexible in others.

❹ **Bi-gram shift (BiShift).** Introduced by (Conneau and Kiela, 2018). A word bi-gram is a tuple of two consecutive words in a sentence. E. g. the word bi-grams of the sentence *'The cat sat on the mat'* are given by `(The, cat)`, `(cat, sat)`, `(sat, on)`, `(on, the)` and `(the, mat)`. The main idea of this task is to randomly pick two adjacent words and change their order. A modified version of the sentence could be *'The sat cat on the mat'* which a probing task classifier should classify as violating legal word orders. A somewhat related task called WO (word order) was proposed by (Adi et al., 2017) in which a classifier is given a sentence embedding as well as the word representations of two words w_1 and w_2. Based on the input, the classifier has to tell, if w_1 comes before w_2 or vice versa.

❺ **Subject-verb agreement (SVAgree).** For a sentence to be considered grammatical, it is import that subject and verb agree with each other. E. g. in the English present tense, an 's' is appended to the verb to denote the third person singular indicative (*'I walk'* vs. *'He walk**s**'*). (Linzen et al., 2016) originally leveraged the notion of subject-verb agreement while studying the capabilities of RNNs to capture long-range dependencies. This idea can be used in our evaluations as well: We purposefully destroy the subject-verb agreement in given sentences and train a classifier to decide whether the sentence is intact or not (an alternative is to test for **subject-verb distance**). This task requires syntactic knowledge to a large extent and is not at all trivial, since it is very difficult to automatically determine which subject the verb refers to. A major problem in this context pose intervening nouns, so-called **agreement attractors**, which happen to be close the the verb but are actually not the subject it refers to. (Bock and Miller, 1991) introduced the term **agreement attraction error** for situations in which a noun is erroneously considered the subject of a verb. A large variety of other tasks focusing on similar relationships can be designed, e. g. **number agreement**, **gender agreement** or **noun-adjective agreement**.

❻ **Top constituent (TopConst).** A sentence can be split recursively into a series of phrase constituents. Consider the example sentence *'John hit the ball'*. The corresponding parse tree is depicted in figure 11. The figure shows that the sentence can be decomposed into $[[\text{John}]_{\text{NP}}[[\text{hit}]_{\text{V}}[[\text{the}]_{\text{Det}}[\text{ball}]_{\text{N}}]_{\text{NP}}]_{\text{VP}}]_{\text{S}}$. The top constituent in this example is `NP VP`. (Shi et al., 2016) labeled sentences with their top constituents. For that, they defined 20 classes.

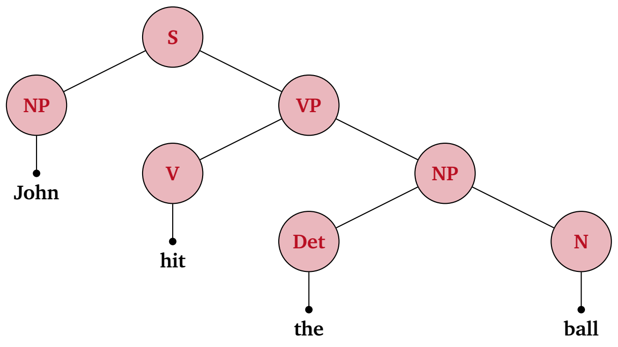


Figure 11: A simple parse tree for the sentence *'John hit the ball'*. Image taken and adapted from: `http://www.wikiwand.com/fr/Arbre_syntaxique` (retrieved: September 05, 2019).

❼ **Tree depth (TreeDepth).** The task introduced by (Conneau and Kiela, 2018) also involves syntax trees as shown in figure 11. In this task the classifier has to predict the length of the longest path (the depth of the tree) from the root note S to the leaves. In the example above the tree depth is equal to 4.

8 **Voice (VOICE).** Many languages offer the **possibility of exchanging subject and object in a sentence**, e. g. *'The police$_{subj}$ deals with the criminal$_{obj}$'* vs. *'The criminal$_{subj}$ is dealt with by the police$_{obj}$'*. The first sentence is said to be in *active* voice, while the second one in *passive* voice. (Shi et al., 2016) trained a Neural machine translation (NMT) system and took the trained encoder to produce sentence representations. These sentence embeddings were classified into either active or passive voice. The trained classifier could detect the correct voice label in 92.8 % of the cases. (Krasnowska-Kieraś and Wróblewska, 2019) pick up this task to test English and Polish sentence embeddings (they call the task PASSIVE).

5.2.3 Probing Tasks for semantic Information

Unlike syntactic tasks, their semantic counterparts do not involve the structure of sentences, they rather test sentence representations for **aspects of meaning or how sentences are interpreted and understood**. The following tasks exist:

9 **Replacement tasks.** A large amount of *'replacement tasks'* has been proposed in the literature. In such tasks, specific words are replaced and a classifier has to predict **whether the sentence is intact or whether it was modified**. These tasks include for example:

- **Antonyms.** (Naik et al., 2018) pick words from the sentence and replace them by antonymous words using *WordNet* (Miller, 1995).
- **Semantic Odd Man Out (SOMO).** Introduced by (Conneau et al., 2018a). In this task, a noun or a verb is replaced without destroying the grammaticality of the sentence. Furthermore, Conneau and colleagues ensure that 'the bigrams formed by the replacement with the previous and following words in the sentence have frequencies that are comparable with those of the original bigrams.'
- Further replacement tasks introduced by (Kim et al., 2019): **Articles.** Definite (*'the'*) and indefinite articles (*'a'*, *'an'*) are exchanged. **Comparative.** Replacement of comparative words like *'better'* or *'worse'*. **Coordinating conjunctions.** Replacement of conjunctions like *'and'*, *'but'*, *'or'*. **Prepositions.** Replacement of prepositions by one another. **Quantification.** Test embeddings for quantifier (words like *'all'*) replacement. **Spatial expressions.** Replacement of spatial words like *'under'* or *'over'*. **Wh-words.** Replacement of question words like *'what'*, *'where'* and *'who'*.

10 **Coordination inversion (COORDINV).** (Conneau et al., 2018a) randomly change the order of coordinate clauses. The task is to decide whether or not the clauses are in their original order. An example from their paper is: *'They might be only memories, but I can still feel each one'* vs. *'I can still feel each one, but they might be only memories'*.

11 **End of sentence (EOS).** Introduced by (Kim et al., 2019). A classifier is given a sentence representation of two concatenated sentences and an integer index. **The task is to assess whether the integer number is a valid index to split the two sentences.** Punctuation is removed and all words are transformed to lowercase in order not to give the classifier any additional hints where the two sentences have to be split. The authors present the following example ('//' denotes the splitting point): ✓ *'the forehead is gathered in a frown // the mouth is slightly parted to reveal the teeth'* vs. ✗ *'the forehead is gathered in a frown the mouth // is slightly parted to reveal the teeth'*

12 **Negation (NEG).** (Naik et al., 2018) construct sentence pairs comprising the original version and a negated version. The two cases have to be told apart. Linguists refer to this as **positive or negative polarity**. Some languages have special words which are inserted into the sentence to negate their meaning while others have affixes which are prepended or appended to words. The Turkish language is an instance of the latter case. (Sahin et al., 2019) give an example for this: *'Dün okul-a git-**me**-di-m.'*[23] where the affix *'me'* negates the verb. Embeddings learning such features are considered good features in sentiment analysis.

13 **Numerical reasoning (NUMREASON).** (Naik et al., 2018) pick sentences containing numbers, subsequently alter and prefix them with expressions like *'more than'* or *'less than'*. The authors present the following example: The sentence *'Tim has 350 pounds of cement in 100, 50, and 25 pound bags'* is changed to *'Tim has less than 750 pounds of cement in 100, 50, and 25 pound bags'*. It is the job of the classifier to classify the two sentences as related.

[23] English translation: *'Yesterday I did not go to school.'*

14 **Semantic role (SemRole).** This task provides information about the extent to which semantic roles (entity-event relationships) are captured by sentence embeddings. For the data set, (Ettinger et al., 2016) label sentences with the entity-event relationship contained in them. E. g. the sentence *'The dog jumps over the fence'* is labeled with its entity-event relationship *'dog-jump'*.

15 **Spelling error (SpellErr).** Introduced by (Naik et al., 2018). This task served as the only noise test in the suite of stress test data sets. The goal of the authors was to show in how far NLI models are robust with respect to spelling errors in the sentences. The data set consists of error-free as well as erroneous sentences which are labeled accordingly. A very similar task which we thought of is to test sentence embeddings for grammaticality. Sentences are altered in a way such that they are no longer grammatical. A classifier is then trained to distinguish between grammatically correct and incorrect sentences.

16 **Subject number/object number (SubjNum / ObjNum).** The concept of grammatical number is an integral part in most languages. In English and many other languages, grammatical number is represented by *singular* and *plural*. (Conneau et al., 2018a) decided to probe sentence embeddings for this linguistic property. To excel in this task, a classifier has to correctly predict the number of the subject. An alternative is to probe for the **object number** which works accordingly.

17 **Tense (Tense).** Introduced by (Conneau et al., 2018a). Sentences are labeled with the tense of their main verb. Given a vector representation of the sentence, the goal is to predict the correct tense. The task is often restricted to present tense and simple past, since other tenses like future, present perfect or past perfect are **compound forms** and most tree banks use labels on word level and do not consider larger constructs. Consider the example sentence *'I will do it.'*: Each word in the sentence is labeled separately, rather than labeling the entire compound *'will do'* as future tense.

5.3 Presentation of Target Languages

The following sections provide a short introduction into the main grammatical characteristics of the target languages. Basic knowledge about these languages is an essential prerequisite when it comes to the evaluation of the probing tasks with respect to their implementability in the target languages (cf. 5.4 on page 27). Some grammatical concepts which are covered by the tasks presented might not be existent in some of the languages. A presentation of English and German is omitted, the focus is rather on the lower-resource languages which may not be as familiar to the English-speaking reader.

5.3.1 Russian Language – русский язык

According to (Reuther, 2009), the number of Russian native speakers is estimated to approximately 145 million people. Furthermore, about 80 million people learned Russian as a second language. The Slavic language is written in **Cyrillic script** (cf. figure 12 for an example). The script consists of 33 letters and distinguishes between upper and lower case letters.

Exegi monumentum

Я памятник себе воздвиг нерукотворный,
К нему не зарастет народная тропа,
Вознесся выше он главою непокорной
Александрийского столпа.

Нет, весь я не умру — душа в заветной лире
Мой прах переживет и тленья убежит —
И славен буду я, доколь в подлунном мире
Жив будет хоть один пиит.

Figure 12: Two stanzas from the poem *'Exegi monumentum'* by Alexander Puschkin. The example was taken from https://www.russlandjournal.de/russische-literatur/gedichte-von-alexander-puschkin/ (retrieved: September 06, 2019).

The Russian language is first of all characterized by **strong inflection** and is therefore a morphologically rich language: Words are adapted to their grammatical function by adding affixes. Word stems can also change entirely. The concept of grammatical **gender is existent** in the Russian language (three distinct genders) but there are **no articles** (neither definite nor indefinite articles). Instead, the gender of a noun has to be read off from the word itself. Nouns are also the group of words which are the most diverse in terms of possible word forms: The case system comprises **six different cases**: ❶ Nominative, ❷ genitive, ❸ dative, ❹ accusative, ❺ instrumental and ❻ prepositional. Verbs do not have as many forms as e. g. in German: There are only **three tenses** (present, past and future tense) and also the word order is not as strict as it is in the German language. Finally, adjectives have to agree with nouns to which they refer in terms of gender and number.

5.3.2 Turkish Language – Türkçe

(Karakurt, 2006) provides an overview of the Turkish language which is also known as Istanbul Turkish in order to distinguish it from other Turkic languages. It is the native language of approximately 65 million people. The Turkish alphabet was introduced 1928 by Mustafa Kemal Atatürk and comprises 29 characters (cf. figure 13 for an exemplary text). It replaced the previously used Arabic script. The language is characterized by **extensive usage of agglutination**. Agglutination refers to the grammatical concept of adding several morphemes to words to adjust them according to their grammatical function in the sentence. This way words can become very long and it is even possible to express entire sentences as one word. Consider the following extreme (and artificial) example: *Çekoslovakyalılaştıramadıklarımızdanmışsınızcasına* which means literally: *'In the manner of you being one of those that we apparently couldn't manage to convert to Czechoslovakian'*.[24]

Orada bir köy var
uzakta
o köy bizim köyümüzdür
gitmesekte
görmesekte
o köy bizim köyümüzdür.

Figure 13: A short poem written in the Turkish alphabet. The example was taken from: http://www.schule-mehrsprachig.at/fileadmin/schule_mehrsprachig/redaktion/sprachensteckbriefe/pdf/ssb_tuerkisch_neu.pdf (retrieved: September 06, 2019).

Turkish sentences usually follow the word order `subject-object-verb` (the verb follows the object, unlike in German). This is the case not only for affirmative clauses but also for interrogative clauses. Furthermore, there is **no grammatical gender** and **neither definite nor indefinite articles** are available. The Turkish language is also characterized by a great extent of regularity. There are only a few exceptions to rules. E. g. verbs usually follow the standard rules for conjugation (only few irregular verbs exist). Noun declension is diverse: The Turkish language differentiates between **six cases**: ❶ Nominative, ❷ genitive, ❸ dative, ❹ accusative, ❺ ablative (used to express motion away from something) and ❻ locative (used to refer to a location). In general, **adjectives are not declined**, i. e. adapted to gender or number.[25]

5.3.3 Georgian Language – ქართული ენა

(Chapidze-Morgenroth, 2015) introduces main aspects of the Georgian language.[26] It is spoken by approximately 4.5 million native speakers and is therefore considered a **low-resource language** in the NLP community.[27] It is a South Caucasian language with its own script called **Mkhedruli.**[28] The alphabet consists of 33 characters (like Cyrillic) each of which corresponds to a phoneme. Furthermore, there is no distinction between capital and non-capital letters. To get a better idea of what the script looks like, figure 14 on the following page shows the beginning of the Georgian national epos *'The knight in the Tiger's skin'* written in Mkhedruli script. Agglutination as well as inflection are typical of the language. **It does not possess the concept of grammatical gender** and also (definite and indefinite) **articles are missing**. Prepositions are replaced by **postpositions**[29] and the Georgian **word order is relatively free**. (Makharoblidze, ?) give an example for this using the sentence *'I am in Tbilisi.'*. The following variants are possible:

[24] The example was taken from: https://en.wikipedia.org/wiki/Turkish_language (retrieved: September 06, 2019).
[25] Cf. https://www.colanguage.com/turkish-adjectives (retrieved: October 02, 2019)
[26] Additional information taken from https://deacademic.com/dic.nsf/dewiki/510990; (retrieved: October 02, 2019)
[27] A language is said to be low-resource, if there is only little annotated training data available that can be used to train NLP systems.
[28] Derived from the Georgian word *'mkhedari'* which corresponds to the English word *'knight'*.
[29] A German example for a postposition would be 'den Fluss **entlang**' (https://de.wikipedia.org/wiki/Georgische_Sprache; retrieved: September 06, 2019).

- me var tbilisshi. (I am Tbilisi-in)
- me tbilisshi var. (I Tbilisi-in am)
- var me tbilishsi. (am I Tbilisi-in)

The language developed a comprehensive case system. It has **seven cases** which are as follows: ❶ Nominative, ❷ ergative, ❸ dative, ❹ vocative (used to address someone), ❺ adverbial (used to mark adverbial phrases), ❻ genitive and ❼ instrumental (corresponds to the preposition 'with' in English: *'He is eating* ***with a fork****'*). While most European languages organize verb forms by tenses, modes and aspects, the Georgian language introduces the **concept of screeves**. There are **eleven screeves** each of which represents a combination of tense, mode and aspect. A screeve does not only affect the verb but also the form and case of the subject and the object. Equally striking is the high number of irregular verbs in the language.[30]

ᲕᲔᲤᲮᲘᲡ ᲢᲧᲐᲝᲡᲐᲜᲘ

ᲨᲝᲗᲐ ᲠᲣᲡᲗᲐᲕᲔᲚᲘ

ღმერთსა შემვედრე, ნუთუ კვლა დამხსნას სოფლისა შრომასა,
ცეცხლს, წყალსა და მიწასა, ჰაერთა თანა შრომასა;
მომცნეს ფრთენი და აღვფრინდე, მივჰხვდე მას ჩემსა ნდომასა,
დღისით და ღამით ვჰხედვიდე მზისა ელვათა კრთომასა.

მზე უშენოდ ვერ იქმნების, რათგან შენ ხარ მასა წილი,
განაღამცა მას ეახელ მისი ეტლი, არ თუ წბილი!
მუნა გნახო, მადვე გსახო, განმინათლო გული ჩრდილი,
თუ სიცოცხლე მწარე მქონდა, სიკვდილიმცა მქონდა ტკბილი!

Figure 14: Two stanzas from *'The knight in the Tiger's Skin'* (Georgian national epos) written in Mkhedruli script. The example was taken from http://ftp.gwdg.de/pub/ctan/fonts/georgian/mxedruli/mxeddoc.pdf (retrieved: September 06, 2019).

The following table 6 summarizes the main aspects for all languages considered in the context of this thesis:

Property	**EN**	**DE**	**RU**	**TR**	**KA**	**Comments**
Gender	✗	✓	✓	✗	✗	
Articles	✓	✓	✓	✗	✗	
Number of cases	✗	4	6	6	7	
Number of tenses	12	6	3	4	11	English: + aspect tenses (-*ing*-forms)
Lower/Upper case letters	✓	✓	✓	✓	✗	
Script	L	L	C	L	M	L: Latin, C: Cyrillic, M: Mkhedruli
Number of characters	26	30	33	29	33	German: + 'ä', 'ö', 'ü' and 'ß'
Inflection/Agglutination	I	I	I	A	I/A	I: Inflectional, A: Agglutinative

Table 6: Summary of the characteristics of the languages presented.

[30] For more information cf. https://deacademic.com/dic.nsf/dewiki/510990 (retrieved: October 02, 2019).

After having introduced the probing tasks as well as the main grammatical aspects of the low-resource target languages (Russian, Turkish and Georgian), we now evaluate the probing tasks with respect to their implementability in the languages considered. As already mentioned, concepts that are available in some languages do not necessarily have to be included in all languages.

One aspect which stands out immediately is that there are **no definite and indefinite articles** in Russian, Turkish as well as in the Georgian language. As a consequence, the ARTICLE probing task is not reasonable for these languages. Furthermore, the word order in Russian and Georgian are not as strict which might cause problems in the WO task as well as in the BISHIFT task. Nevertheless, these probing tasks are still considered possible candidates for the mentioned languages, since it is assumed that there is still a preferred word order which sounds the 'most natural' for native speakers.[31] One further aspect to keep in mind is that **the Georgian language does not have tenses as they are known in European languages**. Instead, the language distinguishes between several 'screeves' (combinations of tenses, modes and aspects) also known as series which adds more complication. This, too, has to be reflected in the framing of the TENSE probing task, if this task is chosen for the Georgian language. Finally, there are **no prepositions in Georgian**. Thus, for a PREPOSITION probing task we would have to replace postpositions.

Agglutination is another aspect which might cause problems in some tasks. One task where this could be problematic is the WC task. Since words rarely appear in their base form in agglutinative languages, it might be difficult to generate large data sets. In such languages, the way the word is used in the sentence changes the surface form of the word by adding affixes to the word stem. Consider an example from the Turkish language:[32] *'evler'* (houses) ⟷ *'evlerim'* (my houses). Whereas in English, the word *'houses'* keeps its surface form, it changes in Turkish.[33]

The following three tables (cf. table 7 on the following page, table 8 on page 29 and table 9 on page 30) summarize the assessments of the implementability of the probing tasks in the column `'Applicability'`. Tasks which are applicable to the language without restriction are marked in `green color`, tasks which are meaningful in general, but the idiosyncrasies of the language might cause problems in the implementation are marked with `orange` color. If a task is not applicable whatsoever (e. g. due to reasons listed above), it is marked in `red` color.

The other column named `'Data creation'` provides information about how easy or difficult it is to create appropriate training data for the probing tasks. A `green` cell symbolizes that the process of creation is easy and can be automated entirely. This is mainly the case for the tasks which probe sentences for surface information. For them, no additional knowledge about the languages is necessary and the data sets can be created by looking at the surface forms of the sentences alone while the meaning or the syntax of the sentences can be disregarded entirely. An `orange` cell denotes that more linguistic knowledge or special NLP tools like taggers or parsers (or tagged data sets and tree banks) are required. Data set creation can be automated to a large extent, if these resources exist, but still, we expect that the implementation entails more problems. A probing task marked with `red` color is very difficult to implement, since resources probably do not exist or the required linguistic knowledge is not available. For example, the NUMREASON task is (almost?) impossible to implement, if one is not able to speak the corresponding language. Furthermore, a manual creation of the data set is probably inevitable. We believe that if the process is partially automated, it is then at least necessary to conduct a thorough visual inspection, since a computer can easily introduce ungrammatical sentences. Finally, a `black` marker refers to the fact that the probing task is not applicable to the language and therefore a sensible data set cannot be created.

The last column called `'Comments'` leaves room for additional remarks and examples.

[31] In the overview provided by (Makharoblidze, ?), the author explains that the basic structure follows `subject-verb-object`, but deviations are allowed.

[32] The example was taken from: `https://www.grammatiken.de/tuerkische-grammatik/tuerkisch-agglutinierende-sprache-suffix-anhaengen.php` (retrieved: October 11, 2019).

[33] Furthermore, both Turkish words would be assigned different word embeddings.

Probing task	Short description	Applicability					Data creation					Comments
		EN	DE	RU	TR	KA	EN	DE	RU	TR	KA	
*Probing tasks for **surface** information*												
Sentence length	Predict the binned length of a sentence	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Length mismatch	Test if two sentences have equal length	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Word content	Decide if a word occurs in the sentence	✓	✓	✓	✓	✓	✓	✓	✓	✻	✻	
Word overlap	Given two sentences, decide if they overlap	✓	✓	✓	✓	✓	✻	✻	✻	✻	✻	Difficult to define overlap
*Probing tasks for **syntactic** information*												
Bi-gram shift	Predict if sentence has legal word order	✓	✓	✻	✓	✻	✓	✓	✓	✓	✓	*kids play* vs. *play kids*
Noun pronoun agreement	Check if noun and pronoun agree	✓	✓	✓	✓	✓	✻	✻	✻	✻	✻	
Subject verb agreement	Check if subject and verb agree	✓	✓	✓	✓	✓	✻	✻	✻	✻	✻	*He goes* vs. *He go*
Top constituent	Test for top constituency of sentences.	✓	✓	✓	✓	✓	✻	✻	✻	✻	✻	`ADVP NP VP .`
Tree depth	Test for hierarchical structure of sentences.	✓	✓	✓	✓	✓	✻	✻	✻	✻	✻	
Voice	Probe for active or passive voice	✓	✓	✓	✓	✓	✻	✻	✻	✻	✻	*drive* vs. *is driven*
Word order	Predict if w_1 is before w_2 or vice versa	✓	✓	✻	✓	✻	✓	✓	✓	✓	✓	

Table 7: Overview of the implementability of the surface and syntactic probing tasks in different languages.

Probing task	Short description	Applicability					Data creation					Comments
		EN	DE	RU	TR	KA	EN	DE	RU	TR	KA	
*Probing tasks for **semantic** information*												
Antonyms	Check if two sentences are antonymous	✓	✓	✓	✓	✓	✻	✻	✻	✻	✻	
Articles	Probe if (in-)definite articles were substituted	✓	✓	✗	✗	✗	✓	✻	■	■	■	RU, TR, KA: No articles
Comparative	Probe for comparative replacement	✓	✓	✓	✓	✓	✻	✻	✻	✻	✻	
Coordinating conjunctions	Probe for conjunction replacement	✓	✓	✓	✓	✓	✻	✻	✻	✻	✻	
Coordination inversion	Predict if coordinate clauses are inverted	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	
End of sentence	Decide where to split two sentences	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Grammaticality	Test if sentence is grammatical	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Negation	Test if sentence is negated	✓	✓	✓	✓	✓	✻	✻	✻	✻	✻	
Numerical reasoning	Test if two sentences contain similar quantities	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	
Object number	Predict the number of the subject	✓	✓	✓	✓	✓	✻	✻	✻	✻	✻	
Preposition	Probe sentences for preposition replacement	✓	✓	✓	✓	✻	✻	✻	✻	✻	✻	Postpositions in KA
Quantification	Test sentence for quantifier replacement	✓	✓	✓	✓	✓	✻	✻	✻	✻	✻	

Table 8: Overview of the implementability of the semantic c probing tasks in different languages (part 1).

Probing task	Short description	Applicability					Data creation					Comments
		EN	DE	RU	TR	KA	EN	DE	RU	TR	KA	
Semantic role	Predict if sentence contains semantic role	✓	✓	✓	✓	✓	*	*	*	*	*	
SOMO	Replace noun by other noun. Predict if sentence is intact	✓	✓	✓	✓	✓	*	*	*	*	*	
Spatial expressions	Test for substitution of spatial words	✓	✓	✓	✓	✓	*	*	*	*	*	***On/Under** desk*
Spelling error	Test if sentence contains spelling errors	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	*He goes to schol*
Subject number	Predict the number of the subject	✓	✓	✓	✓	✓	*	*	*	*	*	*I go* $\rightarrow$ `singular`
Tense	Predict the tense of the sentence	✓	✓	✓	✓	*	*	*	*	*	*	KA: screeves
Wh-words	Decide if a wh-word was replaced	✓	✓	✓	✓	✓	*	*	*	*	*	

Table 9: Overview of the implementability of the semantic c probing tasks in different languages (part 2).

5.5 Implementation of Probing Tasks

The pool of probing tasks already introduced in the literature is vast (cf. section 5.2 on page 21 for more details). For reasons of time, it is impossible to implement all of them and as section 5.4 on page 27 shows, not all probing tasks are equally suitable for all target languages taken into account in this work. Furthermore, some probing tasks (above all those focusing on semantic aspects) **require a significant amount of linguistic understanding of the languages**. Unfortunately, this knowledge is quite limited for the low-resource languages like Russian, Turkish or Georgian and the availability of sophisticated NLP tools and resources such as taggers, parsers or pre-tagged tree banks might also be restricted to higher-resource languages (or might at least not be available to the same extent in low-resource languages). These circumstances necessitate considering a **meaningful subset of the probing tasks** for which enough data is available. Furthermore, the set of tasks should probe for interesting linguistic properties in the target languages and should not be too expensive to conduct in terms of time as well as computational cost. Hence, section 5.5.1 first of all reviews possible corpora which can be used to establish a basis for the data sets. Subsequently, section 5.5.2 on page 34 gives some insight into which probing tasks were chosen and lists reasons for their selection.

The next aspect which has to be given adequate thought is how the tasks are set up, **i. e. how each probing task is framed**. Many possibilities are conceivable in this context: A task may be formulated as a classification problem, but also regression would be possible if the target variable was continuous. In this work the focus will be on classification tasks, since there exist no truly continuous output variables.[34] The explanations in section 5.5.3 on page 35 provide detailed information on how the probing data sets were acquired. Ultimately, section 5.5.4 on page 40 presents how the classification algorithm and its architecture was chosen.

5.5.1 Selection of Tree Bank Corpora

The choice of the data source for the purpose of data set creation is essential. First of all, we notice that **a large number of the probing tasks require knowledge about syntax**. For example in the SVDist task it is crucial to know which word plays the role of the subject and which word is considered the predicate of the sentence. While it is easy to determine these parts of a sentence for the English language due to it being spoken by a great majority, it turns out to be quite difficult for languages like Georgian which is not spoken by most people (including the author of the present thesis). Even if one has profound language proficiency, the sheer number of instances a suitable data set requires is prohibitive for manual data creation and labeling, respectively. In general, there are two approaches which are conceivable in order to get hold of the information necessary: ❶ The first option is to use an unannotated data set which is processed by leveraging pre-trained taggers and parsers. Unfortunately, especially for the low-resource languages, it is utmost difficult to find such NLP tools. ❷ A by far more convenient alternative is to acquire a collection of sentences that has already been processed and which is publicly available. Such data sets are called **tree banks**. We chose the latter approach in the context of this thesis.

Typically, a large amount of tree banks exists, especially for high-resource languages like English or German. All of these could serve as a basis to create probing task data sets. The following list shows the search results for different tree bank corpora in the respective languages:

Corpus	**Link**	**Author/Publication**
❶ English		
Penn Tree Bank	Link	(Marcus et al., 1994)
UD English-EWT (English web treebank)	Link	(Silveira et al., 2014)
UD English-ESL (English as a second language)	Link	(Berzak et al., 2016; Yannakoudakis et al., 2011)
UD English-LinES (English-Swedish corpus)	Link	(Ahrenberg, 2007, 2015)
UD English-GUM (Georgetown university multilayer corpus)	Link	(Zeldes, 2017)
UD English-ParTUT (parallel Turin university treebank)	Link	(Sanguinetti and Bosco, 2014a,b)
UD English-PUD (parallel corpus)	Link	(McDonald et al., 2013)

[34] However, the length of a sentence can be regarded as a quasi-continuous variable.

❷ German

TIGER	Link	(Brants et al., 2004)
TüBa-D/Z	Link	(Hinrichs et al., 2000)
Hamburg Dependency Treebank	Link	(Foth et al., 2014)
UD German-GSD	Link	no author found
UD German-HDT (Hamburg dependency treebank)	Link	(Foth et al., 2014)
UD German-PUD (parallel corpus)	Link	(McDonald et al., 2013)
UD German-LIT (German literary history)	Link	no author found

❸ Russian

UD Russian-GSD	Link	no author found
UD Russian-SynTagRus	Link	(Droganova et al., 2018)
UD Russian-Taiga	Link	(Shavrina and Shapovalova, 2017)
UD Russian-PUD (parallel corpus)	Link	(McDonald et al., 2013)

❹ Turkish

UD Turkish-GB (grammar book corpus)	Link	(Çöltekin, 2017)
UD Turkish-IMST (ITU-METU-Sabancı treebank)	Link	(Sulubacak et al., 2016)
UD Turkish-PUD (parallel corpus)	Link	(McDonald et al., 2013)

❺ Georgian

GNC (Georgian National Corpus)	Link; not publicly available
Universal Dependencies	announced, but not yet available

English. One of the oldest and most prominent tree banks for the English language is the *Penn tree bank* corpus which was created in the period of 1989 to 1996. The corpus contains millions of annotated words (Part-of-Speech (PoS)-tagged, constituency parses, etc.) extracted form a diverse range of sources: E. g. news, manuals or transcribed conversations among many more (Taylor et al., 2003). Another treebank corpus which is well-established in the NLP community is the *Universal Dependencies (UD) tree bank*. It is actually not only one corpus, but a set of diverse corpora from many sources. *UD* is an initiative aiming at providing a consistent tagging scheme across many modern languages. According to the website, the project currently comprises approximately 100 treebanks in more than 70 languages.[35] And the number of supported languages increases steadily.

German. Many resources exist for the German language. Next to *UD* tree banks, other data sets including *TIGER* and *TüBa-D/Z* are available. According to the official website[36] the *TIGER* corpus comprises approximately 50k sentences extracted from German newspapers which were semi-automatically tagged. The corpus can be downloaded in two different formats: *NeGra* format and *Tiger XML*. The *TüBa-D/Z* corpus provided by the University of Tübingen, on the other hand, contains around 105k manually annotated sentences again taken from German newspapers.[37] Unfortunately, this corpus cannot be downloaded without registration and is therefore not publicly available. One of the largest tree banks existing for German is the *Hamburg dependency treebank* containing approximately 260k sentences.[38] As a data source the authors used the famous German website `heise.de`. This tree bank corpus is also available as a *UD* corpus but was reduced in size.

Low-resource languages. As expected, it turns out to be **more difficult to locate sets of tagged sentences for the low-resource languages Russian, Turkish and Georgian**. Luckily, there are *UD* corpora for all these languages, except for Georgian (which has already been announced on the official *UD* website[39]). All *UD* tree banks are in general very

[35] `https://universaldependencies.org/` (retrieved: September 07, 2019)
[36] `https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger.html` (retrieved: September 07, 2019)
[37] `https://uni-tuebingen.de/de/134290` (retrieved: September 09, 2019)
[38] `https://corpora.uni-hamburg.de/hzsk/de/islandora/object/treebank:hdt` (retrieved: September 07, 2019)
[39] `https://universaldependencies.org/`, cf. section *'Upcoming UD Languages'* (retrieved: September 07, 2019)

appealing and **exhibit the same structure** which is why parsing the corpora for all languages is connected with less effort and other tree banks are often not publicly accessible or registration is required when using them. Since the *UD* corpora are furthermore **drawn from a multitude of domains** (e. g. news, literature, web texts, etc.) they are **representative of the different styles and aspects of each language**. Therefore, the tree banks released by *UD* are going to be used as a basis for the creation of probing task data sets. The following figure 15 depicts the general structure of the *UD* corpora:

```
# sent_id = weblog-juancole.com_juancole_20051126063000_ENG_20051126_063000-0029
# text = But in my view it is highly significant.

1  But          but          CCONJ  CC    _                                    8  cc         8:cc         _
2  in           in           ADP    IN    _                                    4  case       4:case       _
3  my           my           PRON   PRP$  Number=Sing|Person=1|Poss=Yes|...    4  nmod:poss  4:nmod:poss  _
4  view         view         NOUN   NN    Number=Sing                          8  obl        8:obl:in     _
5  it           it           PRON   PRP   Case=Nom|Gender=Neut|Number=Sing|... 8  nsubj      8:nsubj      _
6  is           be           AUX    VBZ   Mood=Ind|Number=Sing|Person=3|...    8  cop        8:cop        _
7  highly       highly       ADV    RB    _                                    8  advmod     8:advmod     _
8  significant  significant  ADJ    JJ    Degree=Pos                           0  root       0:root       SpaceAfter=No
9  .            .            PUNCT  .     _                                    8  punct      8:punct      _
   ❶            ❷            ❸      ❹     ❺                                    ❻  ❼          ❽            ❾
```

Figure 15: Exemplary sentence taken from the Universal Dependencies English-EWT corpus.

Columns ❶ and ❷ contain the original words and their lemmas, respectively. Columns ❸ and ❹ contain PoS tags using two different tag sets. Very important for the creation of the data sets is the information which is listed in column ❺. These are **morpho-syntactic annotations** of the words. These annotations give information about cases, tenses, moods, numbers, etc. Finally, the dependency annotations are given in columns ❻, ❼ and ❽ where the number in column ❻ is a reference pointer to the regent of the word. Column ❼ on the other hand specifies the type of relationship between regent and dependent.

Often, there are several smaller *UD* corpora available (like English-EWT, English-ESL, English-LinEs). In order to get the maximum number of sentences, **all *UD* corpora which could be found for one language were concatenated** (also combining the respective `train`, `dev` and `test` portions). For Georgian, an alternative corpus had to be acquired since the language is not yet officially supported by *UD*. The *Georgian National Corpus (GNC)* is in general not publicly accessible, but was made available by courtesy of Mr. Paul Meurer from the University of Bergen (`Paul.Meurer@uib.no`) who is one of the researchers involved in the development and maintenance of this corpus. Unfortunately, the `.zip` file could not be read entirely which is why some sentences are missing. The corpus home page[40] gives more information about the project. It is stated that the corpus is still under construction and it contains texts from Old Georgian, Middle Georgian as well as the modern Georgian language. Also transcriptions of spoken language are included into the corpus. Table 10 summarizes the number of sentences contained in each corpus. Surprisingly, the Georgian corpus is significantly larger compared to the *UD* corpora for the other languages.

Corpus	# sentences
***UD* English**	32,004
***UD* German**	188,517
***UD* Russian**	68,100
***UD* Turkish**	8,513
***GNC* Georgian**	990,669

Table 10: Number of sentences contained in the tree banks for each language.

In some cases it can happen that the sentences from *UD* are not sufficient to build probing sets of adequate size. This is especially critical for the Turkish language where only around 8.5k sentences are available from *UD*, even after concatenation of all available corpora and split portions. Additionally, we notice that not all probing tasks rely on the availability of annotated tree banks (e. g. the SentLen task). A lack of data for such probing tasks was tackled by additionally adding sentences from Wikipedia (cf. table 2 on page 10). This step can be considered critical, since some of the sentence encoder architectures were trained on Wikipedia, but it was necessary due to sparsity of data in low-resource languages. **Attempts to minimize biases were made by resorting to Wikipedia sentences only when absolutely necessary.**

[40] `http://gnc.gov.ge/gnc/page` (retrieved: September 07, 2019)

5.5.2 Selection of Probing Tasks

The choice for the implementation of specific probing tasks was mainly dominated by the availability of the information needed to produce the respective data sets. First of all, all surface probing tasks except for LenMismatch and WOverlap could be implemented. We did not implement LenMismatch, since the probing task objective is similar to the SentLen task. Furthermore, the task requires two sentences to be fed into the classifier which violates the criteria for probing tasks introduced by (Conneau et al., 2018a). The latter argument is also the reason why implementing the WOverlap task was avoided. **In general, all probing tasks implemented in the context of the present thesis adhere to the criteria introduced by Conneau and colleagues.**

It was possible to implement the majority of syntactic probing tasks. The NPAgree (noun-pronoun agreement) task was not implemented due to several reasons: The first reason is that the kind of information it probes is very similar to the SVAgree task. Also, **noun-pronoun relationships often exist beyond sentence boundaries**, since it is considered better linguistic style to prevent multiple mentions of the same object (E. g. *'The house fell victim to the flames. [~~The house~~] → **It** was built in 1836.'*). What is more, the information which pronoun belongs to which noun is **not included** in the underlying corpus which prevents the creation of high-quality data sets. Finally, the TopConst task was excluded, since it turned out to be **extremely difficult to parse enough data such that reasonably balanced data sets could be produced**. The diversity of the sentences in the corpora is so large that an insufficient amount of parses with identical top constituents are the consequence. For Georgian, Mr. Paul Meurer could supply some constituency parses, but the number of instances only amounts to approximately 40. This is considered to be too less data such that an interpretation of the results is meaningless.

The implementation of semantic probing tasks is the most challenging. All replacement tasks necessitate the compilation of lists for all languages containing words which can be used as a substitute. For example the antonyms task would require lists of adjective-pairs where both elements have antonymous meanings (unfortunately, the authors of *Attract-Repel* did not open source their linguistic constraints data sets which could be used for that). The lists need to be created manually, since tools like WordNet are not available for all the target languages and presumably, the size of these lists would have to be very large in order for the probing data set to contain a sufficient amount of training data. **This is why such probing tasks were mostly neglected in the context of this thesis.** Instead, we decided to introduce an entirely new task called SVDist (subject-verb distance). This task is inspired by the SVAgree task, but instead of predicting whether subject and verb agree, the task is to predict the number of tokens between those two sentence elements. Furthermore, the ObjNum task is similar to the SubjNum task. It is easier to implement the latter one, since the subject is directly annotated in the *UD* corpora, while there are several types of objects for which no dedicated labels exist. Thus, the decision fell on the SubjNum task. Tasks like NumReason or CoordInv are very complicated to implement, since they require more than just simple modification operations (like flipping the order of two words). As the necessary knowledge about the lower-resource languages is not available and in order to avoid producing low-quality data sets, we chose not to implement these two tasks and restrict the implementations to tasks for which better quality can be assured. Finally, the Tense task would be straightforward to implement, unfortunately the corpora partially contain past tense sentences only which renders a classification pointless. **Summarizing the above explanations, the following probing tasks were chosen to be implemented:**

- Implemented **surface** probing tasks
  - Sentence length (SentLen)
  - Word content (WC)
- Implemented **syntactic** probing tasks
  - Bi-gram shift (BiShift)
  - Subject-verb agreement (SVAgree) ⟶ **new!**
  - Subject-verb distance (SVDist) ⟶ **new!** ***(not implemented for Georgian)***
  - Voice (Voice)
  - Word order (WO)
- Implemented **semantic** probing tasks
  - End of sentence (EOS)
  - Subject number (SubjNum)

SentLen. Following (Adi et al., 2017), we grouped the sentences in bins according to the number of words. The bins were chosen as following: $[1;4],[5;8],[9;12],[13;16],[17;20],[21;25],[26;29],[30;33],[34;55],[56;\infty[$. As an alternative, it is conceivable to predict the sentence length directly. However, the prediction of the exact length would be too demanding and furthermore, very long or short sentences tend to be rare such that the corresponding classes would be underrepresented in the training data set. This can be mitigated by aggregating these rare cases into a single larger class. As can be seen in figure 16, there is still an imbalance between the classes, but it is less severe due to the binning procedure. Furthermore, the figure reveals that German sentences tend to be longer on average compared to the remaining languages.[41]

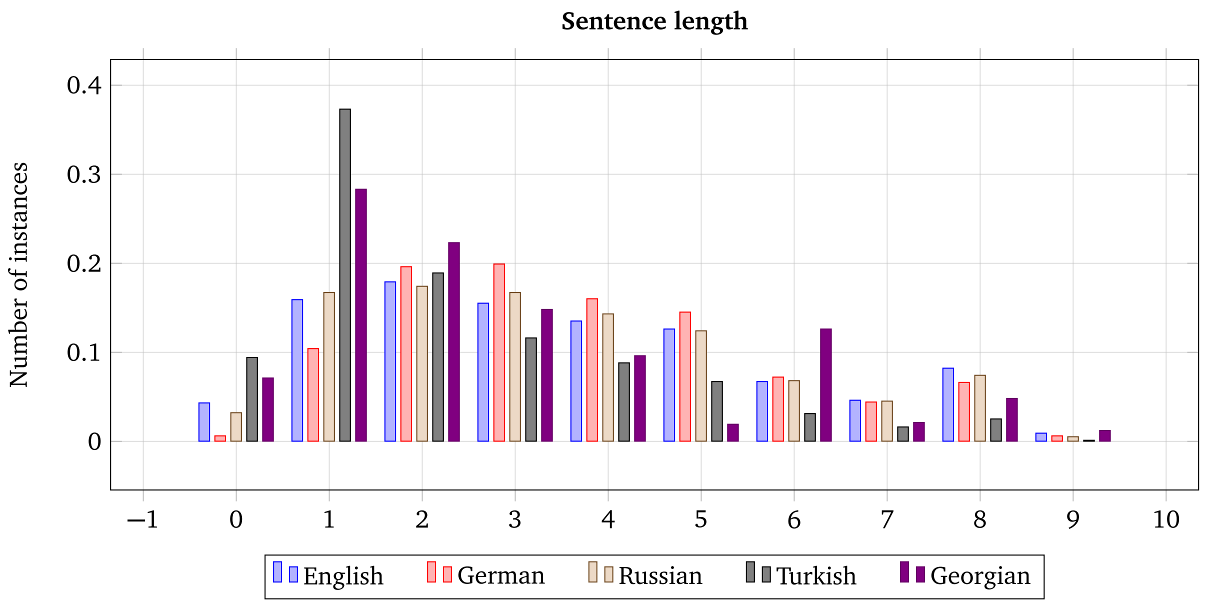


Figure 16: The bar plots depict the sentence distribution by the number of words. As can be seen in the visualization, short to medium-sized sentences (bins 1, 2 and 3) form the majority across all languages. German sentences tend to be longer on average compared to the other languages considered.

	0	1	2	3	4	5	6	7	8	9
English	0.043	0.159	**0.179**	0.155	0.135	0.126	0.067	0.046	0.082	0.009
German	0.006	0.104	0.196	**0.199**	0.160	0.145	0.072	0.044	0.066	0.006
Russian	0.032	0.167	**0.174**	0.167	0.143	0.124	0.068	0.045	0.074	0.005
Turkish	0.094	**0.373**	0.189	0.116	0.088	0.067	0.031	0.016	0.025	0.001
Georgian	0.071	**0.283**	0.223	0.148	0.096	0.067	0.033	0.021	0.048	0.012

Table 11: Distribution of the sentences by their number of words. The majority class is printed in bold face characters.

WC. Since the task setup introduced by (Adi et al., 2017) pollutes the evaluation by introducing additional input to the classifier, we chose to follow the approach of (Conneau et al., 2018a). For this task **it is very important to guarantee that only exactly one word from the list is contained in the sentence**. Otherwise the classification task becomes unfair because one sentence would belong to several classes, thus rendering the class label ambiguous. This circumstance could lead to correct classifications being regarded as misclassifications. Furthermore, due to the binary setup, Adi and colleagues have to construct negative examples, i. e. sentences labeled with an arbitrary word not contained in the respective sentences. The necessity of negative examples is avoided by Conneau's setting.

For the implementation we proceeded as follows: First of all, **instead of sampling 1k words from the corpus, a smaller set of 30 words was picked, because collecting enough training data to distinguish between 1k classes was found to be impossible for the low-resource languages.** As mentioned above, it is crucial that one and only one word from the list is contained in the sentence. In order to facilitate the data generation process, it was therefore necessary to pick words which tend to appear only once in a sentence. Simultaneously their usage must not be that infrequent such that

[41] Due to agglutination it is not surprising that Turkish sentences are shorter on average since a single word contains much more information.

they do appear in sufficiently many sentences. We restricted the choice of words to **30 mid-frequency nouns.**[42] This approach rules out frequently occurring stop words like articles and prepositions which have a tendency to occur more than once in a single sentence. Furthermore, nouns usually appear only once in a sentence, since they are often replaced by pronouns if one wants to refer to them multiple times. Table 29 on page 70 in the appendix gives an overview of the nouns selected for each language.

BiShift. This task probes sentences for legal word orders, where sentences are purposefully manipulated to destroy the legal word order. **We included each sentence twice in the training data set:** The first variant is the original sentence with the correct order of words. The second one is an altered version, where we intentionally switched the order of a word bi-gram. Sentences are mutilated by first splitting the sentence into a series of tokens using the Natural Language Toolkit (NLTK) library – a Python package introduced by (Bird and Loper, 2002) – followed by randomly sampling a word form the sentence and subsequently **switching the position with its right neighbor**. The original sentences are labeled with the positive class (`class 1`), whereas the other half of the sentences belongs to the negative class (`class 0`). Furthermore, 'sentences' in the underlying corpus which comprise less than 2 tokens (which occurs rarely) are omitted.

SVAgree. Probing for the agreement of subject and verb in a sentence requires to construct sentences for which this agreement is broken. We implemented the task as a **binary classification task**, where `class 0` denotes disagreement, while `class 1` represents agreement. The data set was acquired as follows: For each language, we created a list of frequently occurring verbs together with their corresponding present tense conjugations. We checked each individual sentence for the presence of a verb form contained in the list. If no word is present, we excluded the sentence from the probing task data set. **Otherwise, we randomly replace the verb form by a different conjugation in 50 % of the cases.**

The advantage of this approach is obvious: **It does not require annotated corpora and as a consequence larger training sets can be created.** Yet, there are some pitfalls which have to be dealt with in order to acquire high-quality data: ❶ **The first issue is that in some languages, a conjugated verb form may coincide with the infinitive (the base form or lemma) of the verb**. Consider the German language as an example: The first person plural form of a verb is usually identical to its infinitive (at least for regular verbs): *'haben'* ⟷ *'wir haben'*. Also consider the English language which is morphologically less rich. Only the third person singular form is different from the infinitive of the verb. This circumstance is problematic, since it could lead to sentences being included in the data set for which the wrong verb is altered. Let the sentence *'It is nice to* ***see*** *you.'* serve as an example: In this case, the word 'see' is an infinitive and not conjugated. **This means that it is not part of a subject-verb relationship and must not be changed.** In order to avoid a false substitution, all forms coinciding with the infinitive verb form were dropped from the list.

❷ **A second pitfall is the fact that, in the English language, it is often possible to use words as either verbs or nouns** and usually only the context reveals which of the two types it is. For example the word *'drink'* can be used as a verb (e. g. *'He usually drinks sufficient amounts of water.'*) or as a noun (e. g. *'He has already had enough drinks today.'*). Using such verbs in the data creation process can have strange side effects: E. g. the latter sentence would be altered to *'He has already had enough drink today.'* which is equivalent to replacing a plural noun by its singular version. This has nothing to do with subject-verb agreement. Thus, the quality of the data set can be improved substantially by omitting such words. To this end, we used a dictionary to check each word individually whether or not it can be used as a noun. If yes, we dropped the word. Table 30 on page 71 in the appendix gives an overview of the verbs which were chosen for each language.

NB: For Turkish, it was necessary to include more verbs in the list, due to agglutination. Furthermore, it turned out to be difficult to obtain conjugations for Georgian verbs. After conducting extensive research, we found an online presentation[43] which presents frequently used Georgian verbs together with their conjugations. Unfortunately, the format did not allow to copy the words directly. The transliterated forms included in the online presentation combined with a transliteration tool[44] were exploited to obtain the verbs in Georgian script.

SVDist. Next to the subject-verb agreement, we also probe for the distance between subject and verb. **This way we can judge in how far different embedding techniques are capable of capturing long-range dependencies.** In contrast to the SVAgree task, we require an annotated corpus to determine this distance. Fortunately, the *UD* corpora contain dependency parses which can be exploited to implement this task. Figure 17 on the following page depicts an example of such a dependency parse tree.

[42] For the identification of nouns the information contained in column ❹ of the training corpus was used (cf. figure 15 on page 33).

[43] `https://issuu.com/artemivantsov/docs/150_georgian_verbs` (retrieved: September 07, 2019)

[44] `http://ge.translit.cc/` (retrieved: September 07, 2019)

The root of a dependency parse tree is usually given by the main verb[45] of the sentence (word *'was'* in figure 17). In the *UD* tree banks, such words are labeled as `root` in column ❼ (cf. 15 on page 33). The exemplary sentence in figure 15 on page 33 reveals that there are cases where the root element is not given by a verb. We omitted such cases from further consideration. For the rest of the sentences, we saved the row number of the verb and subsequently determined the respective subject by searching for the `nsubj` tag in column ❼. To verify that this word is the subject of the main verb, we used the entry in column ❻. This column contains the row number of the word it refers to. We were able to calculate the distance between subject and verb by taking the absolute value of the difference between the row numbers of the subject and the main verb. In figure 17, the distance between subject (*'delivery'*) and verb (*'was'*) is equal to 1.

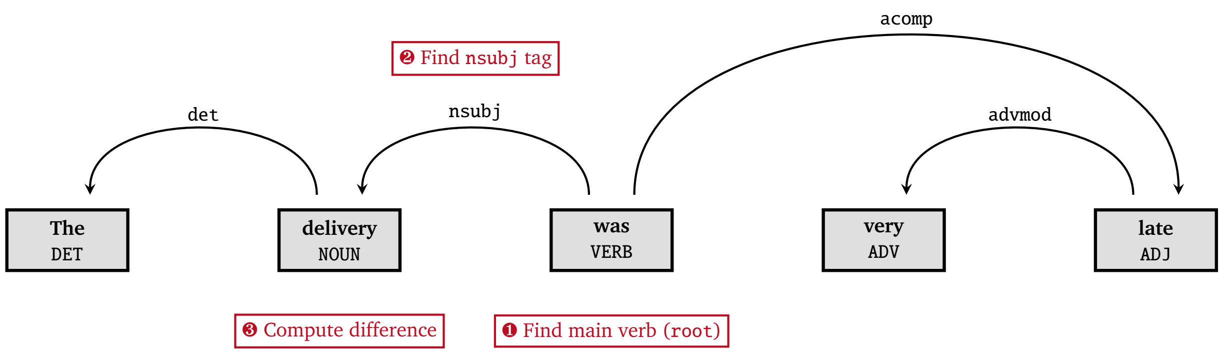


Figure 17: Example of a dependency parse tree. The image was taken and adapted from https://achyutjoshi.github.io/aspect_extraction/aspectextraction (retrieved: October 16, 2019).

Similarly to the SentLen task, predicting the exact distance would be too demanding, which is why a binning procedure using the following buckets was applied: $[1], [2;4], [5;7], [8;12], [13;\infty[$. Because the most frequently occurring case is a distance of 1, we chose to create one dedicated bin for this case. **Unfortunately, this task could not be implemented for the Georgian language due to missing dependency information in the *GNC* tree bank corpus.**

Voice. In order to create probing data, we used the labels listed in column ❺ of the *UD* tree bank corpora. This column contains **morpho-syntactic annotations** which among others include markers indicating *passive* voice (`Voice=Pass`). The *GNC* corpus uses a similar tag. We labeled a sentence with `class 1`, if it contains at least one passive construction. The rest of the sentences was assigned to `class 0` (*active* voice). Unfortunately, we noticed that not all *UD* corpora use the `Voice=Pass` tag, which is why the `aux:pass` tag from the dependency parses (contained in column ❼) were used for English, German, Russian and Turkish. These labels were available for all *UD* corpora. As a consequence, we could generate much more training data which additionally has a better class balance.

WO. We adopted the idea for the task from (Adi et al., 2017), but introduced modifications: For each language, we determined the most frequent noun in the underlying corpora. For all sentences s_i containing this noun, we recorded whether the word appeared at the beginning of the sentence, at the end of the sentence, or in between. Furthermore, we omitted sentences which are too short. **We chose the minimum length of the sentences as well as the boundary to determine in which part of the sentence the word is located in a language-dependent manner, since sentences tend to be shorter in agglutinative languages:**

Language	**Minimum length**	**Begin of sentence**	**End of sentence**
EN, DE, RU	13	tokens 1 to 5	tokens $\text{len}(s_i) - 5$ to $\text{len}(s_i)$
TR, KA	9	tokens 1 to 4	tokens $\text{len}(s_i) - 4$ to $\text{len}(s_i)$

The result of this setting is a **3-way classification task**. In order to make the task more challenging, **we added each sentence three times**: The first variant is the original sentence, whereas the other two variants are random permutations of the words contained in the original sentence, such that the word of interest appears in all three possible positions. We chose the following nouns for the task: EN: *'time'*, DE: *'Jahr'*, RU: 'года', TR: *'zaman'*, KA: ' საქართველოს '.

[45] The main verb is the conjugated verb in the sentence.

EOS. We rephrased the task setting, since the original task introduced by (Kim et al., 2019) is not compliant with the probing task criteria (Conneau et al., 2018a). First, we concatenate two sentences, remove punctuation and convert all words to lowercase (**NB:** Lowercasing was disabled for the German language, since this leads to many OOV nouns). Subsequently, we introduce several segments. Given the embedding of the two concatenated sentences, the classifier has to predict the correct segment where the two sentences have to be split. Thus, instead of the original binary task, we consider a multi-class classification problem. **Again, we chose the segments in a language-dependent manner:**

EN, DE, RU	inflectional	$[1;8],[9;12],[13;16],[17;20],[21;24],[25;28],[29;32],[33;\infty[$
TR, KA	agglutinative	$[1;4],[5;8],[9;12],[13;16],[17;20],[21;\infty[$

SubjNum. We categorize sentences into `class 0`, if their subject is *singular* (`Number=Singular`, cf. column ❺ in figure 15 on page 33), whereas we assign `class 1` if the subject of the sentence is *plural* (`Number=Plural`). Subjects could be identified by the `nsubj` tag in column ❼. **We dropped sentences if their corresponding subject did not have a number tag. We also excluded sentences, if they contained more than one subject, since it is possible that each subject has a different number which renders the classification task ambiguous.**

The following tables 12, 13 on the next page and 14 on the following page show the class distributions for each probing task in the target languages:

English/German									
Classes	SentLen	WC	BiShift	SVAgree	SVDist	Voice	WO	EOS	SubjNum
# inst.	10k/10k	10k/10k	10k/10k	10k/10k	10k/10k	9,999/9,999	10k/10k	10k/10k	9,448/9,999
Class 0	04.3/00.6	01.0/00.6	50.2/49.8	51.1/50.7	43.5/43.6	61.0/50.0	33.3/33.3	21.7/14.4	75.7/50.0
Class 1	15.9/10.4	03.3/02.7	49.8/50.2	48.9/49.3	47.5/28.5	39.0/50.0	33.3/33.3	16.6/18.7	24.3/50.0
Class 2	17.9/19.6	02.4/00.9			04.5/11.5		33.4/33.4	15.5/19.3	
Class 3	15.5/19.9	00.4/02.0			02.7/10.1			13.6/15.5	
Class 4	13.5/16.0	00.9/02.7			01.8/06.2			10.6/11.7	
Class 5	12.6/14.5	02.5/04.1						08.1/08.0	
Class 6	06.7/07.2	00.3/02.0						05.7/05.1	
Class 7	04.6/04.4	01.3/01.0						08.2/07.3	
Class 8	08.2/06.6	03.4/01.0							
Class 9	00.9/00.6	00.5/14.8							
Class 10		04.5/02.4							
Class 11		10.7/06.6							
Class 12		01.8/02.7							
Class 13		00.9/02.4							
Class 14		13.3/03.9							
Class 15		01.7/05.1							
Class 16		00.8/02.2							
Class 17		01.2/02.1							
Class 18		05.0/04.7							
Class 19		00.7/02.5							
Class 20		06.7/01.3							
Class 21		02.7/01.6							
Class 22		01.9/05.2							
Class 23		11.3/02.2							
Class 24		06.2/05.7							
Class 25		00.9/02.3							
Class 26		05.9/08.7							
Class 27		02.0/02.4							
Class 28		03.1/01.5							
Class 29		02.5/03.1							

Table 12: Class distributions for the probing tasks in English and German. One entry is given by 'en/de'.

Russian/Turkish									
Classes	SentLen	WC	BiShift	SVAgree	SVDist	Voice	WO	EOS	SubjNum
# inst.	10k/10k	10k/10k	10k/10k	10k/10k	10k/2,750	10k/8,416	10k/10k	10k/10k	9,999/4,030
Class 0	03.2/09.4	03.8/01.6	48.7/49.5	49.9/50.6	42.3/20.7	68.0/86.2	33.3/33.3	23.6/12.3	50.0/83.4
Class 1	16.7/37.3	05.0/12.6	51.3/50.5	50.1/49.4	40.1/39.0	32.0/13.8	33.3/33.3	17.8/23.7	50.0/16.6
Class 2	17.4/18.9	02.1/00.7			09.7/16.7		33.4/33.4	16.5/18.1	
Class 3	16.7/11.6	02.9/02.1			05.3/11.8			14.1/15.6	
Class 4	14.3/08.8	02.1/02.5			02.7/11.8			10.1/11.3	
Class 5	12.4/06.7	03.1/02.5						07.4/19.1	
Class 6	06.8/03.1	02.7/00.3						04.4/-	
Class 7	04.5/01.6	02.0/06.9						06.1/-	
Class 8	07.4/02.5	06.3/03.9							
Class 9	00.5/00.1	04.6/00.5							
Class 10		03.5/02.5							
Class 11		02.7/03.2							
Class 12		02.9/03.7							
Class 13		01.3/00.2							
Class 14		02.5/03.7							
Class 15		09.7/02.1							
Class 16		03.1/01.4							
Class 17		04.1/02.0							
Class 18		03.6/02.7							
Class 19		05.0/01.7							
Class 20		0.30/01.0							
Class 21		03.8/01.1							
Class 22		02.1/09.7							
Class 23		01.6/12.6							
Class 24		03.0/04.9							
Class 25		02.3/08.8							
Class 26		03.4/01.3							
Class 27		02.7/02.0							
Class 28		02.2/01.7							
Class 29		02.9/00.2							

Table 13: Class distributions for the probing tasks in Russian and Turkish. One entry is given by 'ru/tr'.

Georgian									
Classes	SentLen	WC	BiShift	SVAgree	SVDist	Voice	WO	EOS	SubjNum
# inst.	9,989	10k	9,993	10k	-	10k	10k	10k	-
Class 0	07.0	02.1	51.5	50.8	-	34.9	33.2	26.1	-
Class 1	28.3	01.9	48.5	49.2	-	65.1	33.3	28.2	-
Class 2	22.3	04.0					33.5	13.7	
Class 3	14.8	08.6						08.6	
Class 4	09.6	04.5						06.3	
Class 5	06.7	01.9						17.2	
Class 6	03.3	12.6							
Class 7	02.1	00.2							
Class 8	04.8	09.8							
Class 9	01.2	00.7							
Class 10		00.9							
Class 11		00.3							
Class 12		11.9							
Class 13		01.1							
Class 14		00.3							
Class 15		13.4							
Class 16		01.6							
Class 17		03.8							
Class 18		00.3							
Class 19		02.5							
Class 20		02.8							
Class 21		00.3							
Class 22		02.1							
Class 23		03.9							
Class 24		00.4							
Class 25		01.8							
Class 26		03.5							
Class 27		01.8							
Class 28		00.3							
Class 29		00.5							

Table 14: Class distributions for the probing tasks in Georgian.

5.5.4 Presentation of Classifier Architecture

Researchers who deal with the evaluation of word or sentence embeddings mainly employ two types of machine learning models: **Logistic regression** and **MLPs**. Since neural networks are the state-of-the-art technique at the moment, a similar approach as in (Perone et al., 2018) is taken. **For all probing tasks, a simple feed-forward MLP is trained using the following hyper-parameter setting:**

Number of hidden layers:	1
Number of hidden units:	50
Hidden layer activation:	Sigmoid
Dropout rate:	0.00
Number of epochs:	100
Optimizer:	Adam
Training loss:	Categorical cross-entropy

The input dimensionality depends on the sentence embedding algorithm which is used to encode the sentences, while the number of output nodes is determined by the number of classes considered in each probing task. Furthermore, in order to get a better and more stable estimate of the network's performance, we conduct an n-fold cross-validation approach. **For the evaluation, we chose $n = 5$.**

5.6 Summary

This chapter introduced the concept of probing tasks as well as relevant tasks from the literature (cf. section 5.2 on page 21). (Conneau et al., 2018a) organizes probing tasks for sentence embeddings into three distinct types: Surface tasks, syntactic tasks and semantic tasks. We adopted this categorization. After a brief presentation of the low-resource languages (cf. section 5.3 on page 24), we evaluated all probing tasks w. r. t. their implementability in the target languages (cf. section 5.4 on page 27). The surface probing tasks turned out to be the easiest to implement. Syntactic and semantic probing tasks require more linguistic knowledge about the languages and are harder to implement as a consequence. Finally, section 5.5 on page 31 gave detailed insight into the selection of probing tasks and the subsequent data generation process. For all languages, the sentences we used were taken from the *UD* tree bank corpora as well as Wikipedia, except for Georgian for which we used the *GNC* corpus as a substitute.

We implemented the following list of probing tasks: SentLen, WC, BiShift, SVAgree, SVDist (not for Georgian), Voice, WO, EOS and SubjNum (not for Georgian). Also, the architecture of the probing task classifier was presented briefly (cf. section 5.5.4): For all probing tasks, a simple MLP architecture is employed, where the hyper-parameters were chosen according to (Perone et al., 2018).

6 Downstream Applications

6.1 Introduction

The intrinsic evaluation setting using probing tasks (cf. section 5 on page 21) allows for computationally efficient and fast analysis. Yet it is important to keep in mind that the creation of sentence embeddings in NLP does not end in itself. Embeddings are rather designed to serve as input for higher-order tasks like Named Entity Recognition (NER) or Argumentation Mining, to name just two examples. Hence, the results produced by intrinsic evaluation methods might not necessarily reflect the performance in real applications due to the isolated consideration without any reference to possible fields of application. In the evaluation of word embeddings, (Chiu et al., 2016) could show that good performance in intrinsic tasks does not entail good performance in real NLP applications. The authors conducted intrinsic evaluation using word similarities and correlated the results with the performance in downstream tasks. They found a negative correlation between intrinsic and extrinsic performance which implies that the two methodologies focus on different aspects. On the other hand, (Qiu et al., 2018) concludes that the two approaches are mostly consistent at least for the Chinese language. **Since the research community does not seem to fully understand how results of intrinsic and extrinsic evaluation techniques connect to each other, we include downstream tasks as an additional pointer to representational quality of sentence embeddings.**

The rest of this chapter is organized as follows: Section 6.2 presents ArgMin (argumentation mining) as one downstream task on which the sentence embeddings are tested. Subsequently, the Senti task (sentiment analysis) is outlined in section 6.3 on the following page, before the question type task, also known as TREC, is introduced in section 6.4 on page 44. The final section 6.5 on page 45 introduces details about the classifier architecture which we leverage in the context of this thesis.

6.2 Argumentation Mining

ArgMin task overview. The beginning of argumentation dates back to the philosophers of ancient Greece and is still studied by philosophers today (Moens, 2013). When attempting to convince others of certain statements, people usually give arguments which support their claim in order to increase their credibility. As (Park and Cardie, 2014) noted, contents extracted from online platforms often lack these justifications. People are free to upload statements on the internet (almost) without any restrictions. As a consequence, there is an increasing need to determine whether their claims are justified or not. However, vast amounts of data prevent processing the content manually. Argumentation mining – **the task of automatically finding evidence supporting a claim or refuting it** – is a field in computational linguistics and machine learning which aims to provide a remedy. Automatic argumentation mining has already been done in a multitude of applications, ranging from legal cases (Moens et al., 2007; Palau and Moens, 2011) to online documents (Boltužić and Šnajder, 2014). It is related to opinion mining but **it is less about what people think but is rather concerned with providing reasons for why people think the way they do** (Habernal et al., 2014).

	Abortion	Cloning	Death penalty	Gun control	Marijuana legalization	Minimum wage	Nuclear energy	School uniforms
Number of sentences	3,928	3,038	3,650	3,340	2,474	2,472	3,575	3,007
Total Σ	25,484							

Table 15: The eight topics covered in the sentential argumentation mining data set.

Data set creation. The *UKP argumentation mining* data set used in this context was published by (Stab et al., 2018). The corpus they created comprises sentences connected to **eight different topics**. The sentences either represent an argument supporting a given topic or refuting it. Additionally, non-argumentative sentences were included. The sentences are labeled accordingly, where the labels were obtained using crowd souring on Amazon Mechanical Turk (AMT).[46] The data

[46] https://www.mturk.com/ (retrieved: September 08, 2019)

set is publicly available.[47] We translated the sentences into all target languages. The input to the classifier is given by a concatenation of sentence embedding and topic encoding. The following table 15 on the previous page enumerates the eight topics contained in the data set and shows the number of sentences for each topic:

6.3 Sentiment Analysis

SENTI task overview. Sentiment analysis is referred to as the **automatic detection of sentiment in texts**. The application areas for sentiment analysis are manifold: Ranging from e. g. e-learning applications (Koehler et al., 2015; Ortigosa et al., 2014) to product reviews (Fang and Zhan, 2015). The knowledge about opinions is essential for many groups: Companies are able to modify products according to the customers' desires or political parties can adjust election programs based on voters' sentiment. Sentiment analysis data sets usually contain three classes: One for positive sentiment (sentences labeled as `POS`), one for negative sentiment (`NEG`) and one for neutral sentiment (`NEU`). The following exemplary sentences were taken from the *Twitter US Airlines Sentiment* data set.[48]

`POS` *'@VirginAmerica it was* `amazing` *, and arrived an hour early. You're* `too good` *to me.'*
`NEG` *'@VirginAmerica you guys* `messed up` *my seating... I reserved seating with my friends and you guys gave my seat away...'*
`NEU` *'@VirginAmerica are flights leaving Dallas for Seattle on time Feb 24?'*

Data set creation. Sentiment analysis is a classical NLP application, which is why there exists an abundance of data sets for many languages. For this reason, we chose not to translate an English data set, but to select an original one for each target language. This has the advantage that no translation biases are introduced into the evaluation. The following list shows the results of the search for sentiment data sets ordered by language:

Corpus	**Link**	**Author/Publication**
❶ English		
IMDb movie reviews	Link	(Maas et al., 2011)
US Airline Twitter Sentiment	Link	no publication found
❷ German		
GermEval	Link	(Wojatzki et al., 2017)
SB-10k	Link	(Cieliebak et al., 2017)
DAI Tweet Data Set	Link	(Narr et al., 2012)
German Sentiment Data (Hasso Plattner Institut)	Link	Hasso Plattner Institut Potsdam
❸ Russian		
Russian Twitter Corpus (RuTweetCorp)	Link	Paper (Russian)
Kaggle Russian Sentiment Analysis	Link	no publication found
RuSentiment	Link	(Rogers et al., 2018)
❹ Turkish		
TREMO Data Set	Link	(Alp Tocoglu and Alpkocak, 2018)
Turkish Sentiment Data Set	Link	(Hayran and Sert, 2017)
Turkish Sentiment Analysis Data Set	Link	(Uçan et al., 2016)
Turkish Twitter Sentiment Data	Link	GitHub (no publication found)
❺ Georgian		
None		

[47] https://www.informatik.tu-darmstadt.de/ukp/research_6/data/argumentation_mining_1/ukp_sentential_argument_mining_corpus/index.en.jsp (retrieved: September 08, 2019)
[48] https://www.kaggle.com/crowdflower/twitter-airline-sentiment (retrieved: September 08, 2019)

English. Many data sets are available for the English language. The above enumeration is by far not exhaustive and only lists two of the most prominent data sets in this domain. Both, the *IMDb* data set as well as the *US airline Twitter sentiment* data set are used quite frequently to evaluate classifiers on the task of sentiment analysis. The *IMDb* data set consists of one text file for each movie review which is associated with additional effort in the data reading phase. The *US airline Twitter sentiment* data set, on the other hand, provides the tweets as well as the labels in a single file which permits comfortable data access. For this reason, we chose to use the latter one.

German. For German, four data sets could be detected: The German sentiment data provided by the Hasso Plattner institute in Potsdam comprises only a few hundred examples which is considered insufficient. Furthermore, the official links to download the *DAI Tweet* data set are unfortunately not working (server response: `Service Temporarily Unavailable`), which is why this data set was dropped from the list of candidates. (Cieliebak et al., 2017) published the *SB-10k sentiment* data set which is based on German Twitter data. Unfortunately, the authors only provide tweet message IDs, which means that the tweets themselves have to be downloaded from Twitter separately. For this, a Twitter account and special software modules are required. We decided to use the *GermEval* data set, since it is easily accessible using Pandas and includes sufficient amounts of data.

Russian. We found three Russian sentiment data sets: The *RuSentiment* data set comprises messages extracted from the Russian social network called 'VKontakte'. Due to a request by VKontakte, the data set is no longer publicly available as mentioned in the `readme.md` file on GitHub.[49] Kaggle provides another data set which comes into question. Unfortunately, we could not find a publication which uses this data set. We finally chose the *RuTweetCorp* corpus consisting of approximately 225k examples. It was annotated using only two labels, `POS` and `NEG`. For the use in this work, we abridged the data set to approximately 15k instances.

Turkish. It was difficult to find publicly accessible sentiment data sets comprising enough training examples for the Turkish language. The three data sets authored by (Alp Tocoglu and Alpkocak, 2018; Hayran and Sert, 2017; Uçan et al., 2016) are not publicly available and access to the data has to be registered by filling out respective online forms. We selected a free data set which is available on GitHub.

Georgian. While we were able to find sentiment data for English, German, Russian and Turkish, we could not find an annotated sentiment data set for Georgian. In order to get suitable training data for this language, we use an approach similar to (Choudhary et al., 2018). In their paper the authors perform sentiment analysis for the Indian languages Hindi, Bengali and Telugu and were also confronted with a lack of data. They proceeded as follows:

Positive sentiment		Neutral sentiment		Negative sentiment	
Emoji	**Short name**	**Emoji**	**Short name**	**Emoji**	**Short name**
😍	`:heart_eyes:`	😶	`:no_mouth:`	😢	`:cry:`
😀	`:grinning:`	🤔	`:thinking_face:`	😠	`:angry:`
😁	`:grin:`	😐	`:neutral_face:`	😡	`:rage:`
😂	`:joy:`	😏	`:smirk:`	😭	`:sob:`

Table 16: Emojis and their sentiment. Icons taken from: https://emojipedia.org/ (retrieved: September 18, 2019).

They made use of the Twitter API to download a considerable amount of tweets. The approach of exploiting microblogging platforms as a data source has already been taken quite frequently in the context of sentiment analysis (Agarwal et al., 2011; Pak and Paroubek, 2010). Hundreds of thousands of tweets are posted every day. Thus, one of the first questions that arises is which tweets to collect. **As a first step, the authors created word lists containing the 1k most frequent words in the respective languages.** To this end, they used the website https://1000mostcommonwords.com/ (retrieved: September 08, 2019). Instead of manually labeling all collected tweets with their corresponding sentiment (`POS`, `NEG` and `NEU`), the authors chose a different approach: **They made use of emojis included in the Twitter messages.** These icons carry information about the writers' sentiments. A classification of the most common emojis was made into positive sentiment, negative and neutral sentiment (cf. table 16). Subsequently, the authors queried the Twitter API,

[49] https://github.com/text-machine-lab/rusentiment

where the search string contains one word from the list as well as one emoji. The extracted tweets were then labeled according to the sentiment associated with the emoji. Since some messages contain more than one emoji, tweets can be added multiple times. To avoid this, (Choudhary et al., 2018) used a **hashing mechanism** and dropped messages, if a tweet with the same hash has already been added to the list.

We adopted this approach for the Georgian language. **Additionally, we established a cleaning mechanism to remove content which might impede the analysis.** For example, characters which do not belong the the Georgian script as well as hash tags, user references and web addresses were filtered out. The final data set comprises approximately 11.5k labeled Georgian tweets.

6.4 TREC Question Type Detection

TREC task overview. The *Text Retrieval Conference (TREC)* data set consists of a set of questions which are labeled with their respective question types. For example, the label of the sentence *'What U.S. state is Fort Knox in?'* is given by `LOC` for location. Such data sets are important for the task of question answering where the goal of the system is to automatically find precise answers to given questions by searching for relevant documents and extracting the correct answers from them. To this end, it is essential to determine what kind of answer the question at hand requires (e. g. person, location, description, time). The original TREC data set was created by (Voorhees and Tice, 2000). Further data is provided by (Li and Roth, 2002, 2005) whose data set was also integrated into *SentEval*.[50]

Data set creation. We translated the English data set into the remaining target languages. The labels used in the data set consist of two components. An example of such a label is `LOC:state`. The first part (`LOC`) specifies the general category of the question, while the second part following the colon (`state`) is a subordinate term specifying the type of location more in detail. Since the goal of this work is to evaluate sentence embeddings, rather than to perform question answering, we chose to remove the second part from the label such that only the superordinate category remains (like in *SentEval*). **There are six possible labels:**

1. `ABBR` – abbreviation
2. `DESC` – description
3. `ENTY` – entity
4. `HUM` – human/person
5. `LOC` – location
6. `NUM` – number

The following table 17 summarizes the class distributions in all downstream application data sets considered:

Class distributions							
Classes	ArgMin	TREC	Senti (en)	Senti (de)	Senti (ru)	Senti (tr)	Senti (ka)
# inst	25,303	5,952	14,148	17,908	30,000	6,172	11,513
Class 0	56.4	21.8	15.4	06.2	51.2	30.1	54.8
Class 1	19.4	22.6	64.0	26.0	48.8	17.9	10.0
Class 2	24.2	01.6	20.6	67.8		52.0	35.3
Class 3		21.6					
Class 4		17.0					
Class 5		15.4					

Table 17: Class distributions for the downstream tasks.

[50] https://cogcomp.seas.upenn.edu/page/resource_view/49 (retrieved: September 08, 2019)

6.5 Presentation of Classifier Architecture

All of the downstream tasks introduced above are going to be classified using the following MLP architecture, where the values of the hyper-parameters were again taken from (Perone et al., 2018) (the setup is identical to the probing task classifier):

Number of hidden layers:	1
Number of hidden units:	50
Hidden layer activation:	Sigmoid
Dropout rate:	0.00
Number of epochs:	150
Optimizer:	Adam
Training loss:	Categorical cross-entropy

Again, the cross-validation with $n = 5$ is used for the evaluation. The TREC task poses an exception. We found that the evaluation on the dedicated test split gave results which were much closer to the literature.

6.6 Summary

This chapter gave a brief introduction into the downstream classification tasks which we are going to use in the subsequent evaluation. Section 6.2 on page 41 introduced argumentation mining (ArgMin). The task of the classifier is to decide, if a sentence embedding represents an argument for or against a given topic, or whether it represents no argument at all. Further, section 6.3 on page 42 presented sentiment analysis (Senti) as a second downstream application. The goal is to automatically detect the sentiment in short snippets of texts. Next to the general task introduction, also the corpora we use for this task were presented.

In order to consider a richer and more diverse set of downstream tasks, we included an additional data set from the *SentEval* framework (Conneau and Kiela, 2018), namely the TREC task commonly used in the domain of question answering (cf. section 6.4 on the preceding page). The data set was translated in all target languages. Finally, section 6.5 outlined the architecture of the classifier which is used in all downstream applications. Basically the same architecture is used as in the probing task setup.

7 Evaluation and Results

7.1 Introduction

The previous chapters provided detailed information about the implementation of the probing tasks as well as the downstream applications. After having created and post-processed all data sets needed for the purpose of evaluation, it is now possible to produce results by learning classifiers on top of the sentence embeddings generated from these data sets. This chapter is organized as follows:

Section 7.2 introduces the results obtained for the probing tasks. They were executed for all languages and the resulting classifier performances measured in terms of F1 (and accuracy where required) are reported. The main observations are then briefly analyzed and explained. The subsequent section 7.3 on page 53 proceeds analogously for the downstream applications by presenting the results obtained on these tasks. Again, we highlight important results. In the following section 7.4 on page 56, the probing task and downstream task performances are correlated in order to provide further insights into the connections between these two types of tasks.

During the analysis of the results, we found some discrepancies when comparing them to outcomes reported in the literature. This is why further experiments were conducted to isolate the effects of different setup choices. The WC task serves as an example in the experiments. In a first step, the effects of the classifier architecture as well as the size of the training data set on the relative ordering of the embeddings are examined. In this *'stability analysis'*, data set sizes and classifier architectures are systematically altered and the effects of these changes are recorded. We further investigate factors like class balance of the data set and hyper-parameter tuning. In-depth information on this experiment can be found in section 7.5 on page 60.

7.2 Presentation of Probing Task Results

Figure 18 on the following page depicts the most important results obtained by evaluating the sentence embeddings on probing tasks. Some of the algorithms as well as the exact numbers were left out on purpose in order to facilitate a quick overview and to increase legibility. Please refer to the tables in appendix B on page 72 for detailed information on the probing task results for all five languages. The F1 score was used to measure the performance of the embeddings on the probing tasks, since some of the data sets are imbalanced. Also numbers according to the accuracy metric were produced where needed (cf. appendix B on page 72), given the fact that most of the results in the probing task literature are reported using this metric. The classifier was chosen to be a neural network according to the specification which can be found in section 5.5.4 on page 40. In the following paragraphs we give a textual description of the main observations as well as some explanations:

On the performance of trained encoders. Across almost all languages it is clearly visible that **trained sentence encoders tend to work best**. (Conneau et al., 2018a) find the same pattern for the English language. The superiority of trained sentence embedding architectures is not very surprising because they learn more sophisticated operations on top of word embeddings, instead of simply computing averages. Similarly to (Krasnowska-Kieraś and Wróblewska, 2019), the numbers suggest that ***LASER* embeddings work particularly well on many probing tasks across different languages**. Figure 19 on page 48 shows that for English and German, this embedding is 7× among the top three of the best embeddings. For Turkish, the model even achieves top-three performance 8× (with partially worse absolute scores). (Krasnowska-Kieraś and Wróblewska, 2019) have found no concise explanation for the good performance. We believe that the main reason for this is the multi-lingual setting in which this model was trained. By exploiting similarities among languages, it is easier to produce representations of higher quality. However, the exact reasons require further investigation. **Nevertheless, this superior performance cannot be observed in the results for Georgian**: There, *LASER* does not perform as convincingly as on higher-resource languages in terms of absolute scores (but remains competitive with other Georgian embeddings in terms of top-count performance as can be seen in figure 19 on page 48). While the encoder may have been trained on 93 languages, the training set portion for Georgian was much smaller compared to the other languages due to the low-resource property (KA: 296k ⟷ DE: 8.7m). The fact that trained embeddings perform worse (measured in absolute numbers) in lower-resource languages is a general trend. As can be seen in figure 18 on the following page, **the gap between compositional embeddings and trained sentence encoders shrinks to a great extent**: Averaging methods and random encoders occasionally surpass trained embeddings or perform on par.[51] E. g. *InferSent* sentence embeddings perform rather well on English probing tasks with a healthy gap to average embeddings,

[51] An exception pose probing tasks which explicitly probe for word order information. Due to their nature, compositional models do not have knowledge about word order.

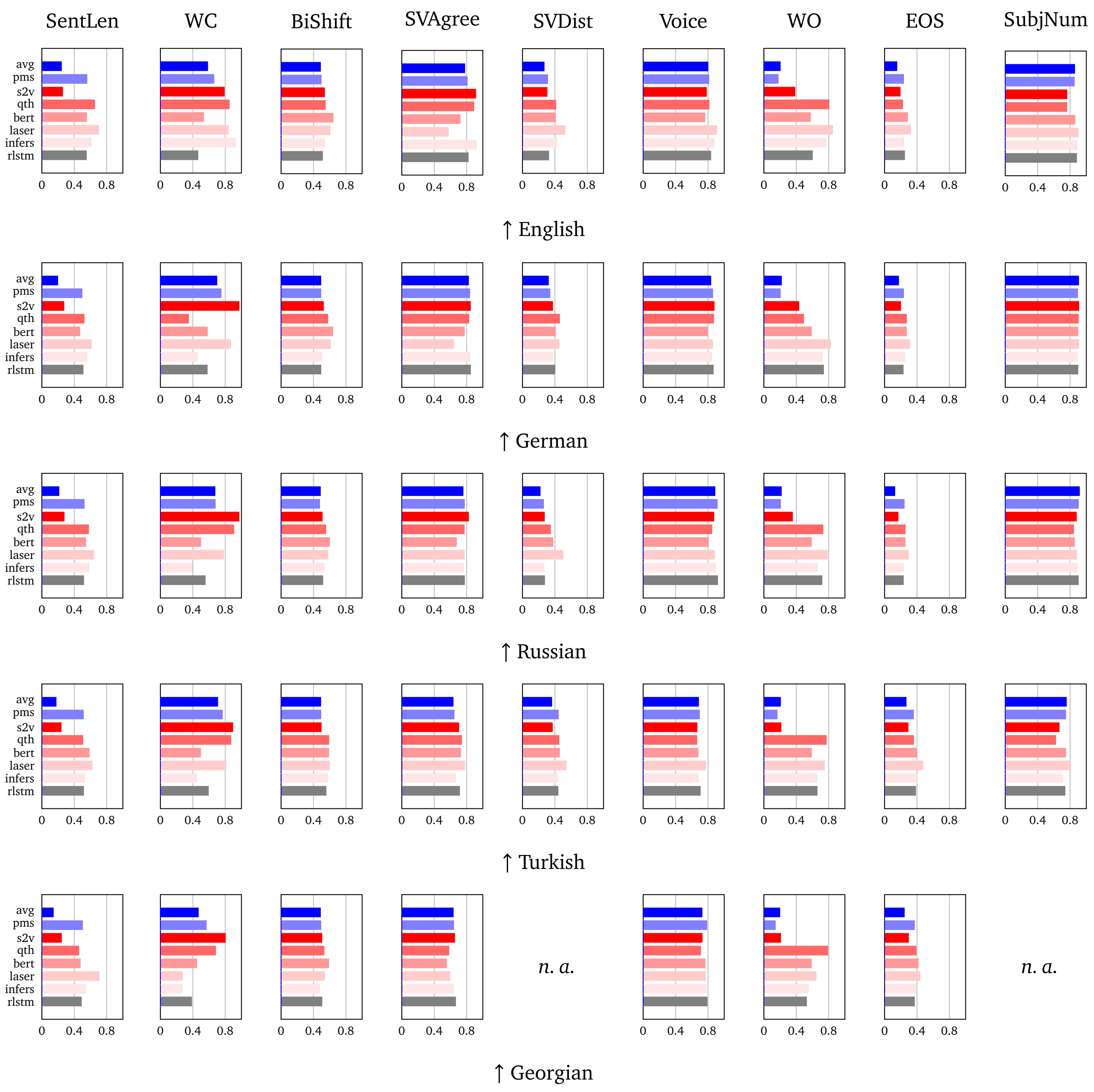


Figure 18: Probing task results for all languages (F1 scores). For reasons of legibility only the most important embeddings are shown where 'avg' stands for *vanilla average*, 'pms' refers to the *concatenated power means* embedding, 'qth' is the abbreviation for *Quick-Thought* and finally, 'rlstm' denotes the *random BiLSTM*). Average embeddings were plotted using a blueish color (all average methods use *FastText* word embeddings) whereas trained sentence encoders are shown in a reddish color. The random BiLSTM encoder serves as a baseline and was printed in gray color. The amount of resources available decreases from top to bottom, thus, Georgian is the lowest-resource language.

but achieve lower scores in other languages. **This decrease in performance is mainly credited to less sentence pairs available for training.** While the English model was trained on the full *AllNLI* data set (*SNLI* + *MultiNLI*), models in low-resource languages had to resort to a reduced amount of *SNLI* sentences due to the bottleneck the Google translator API imposes. For the German model, the entire *SNLI* corpus was available in advance, but the *MultiNLI* data (433k sentence pairs) is missing as well.

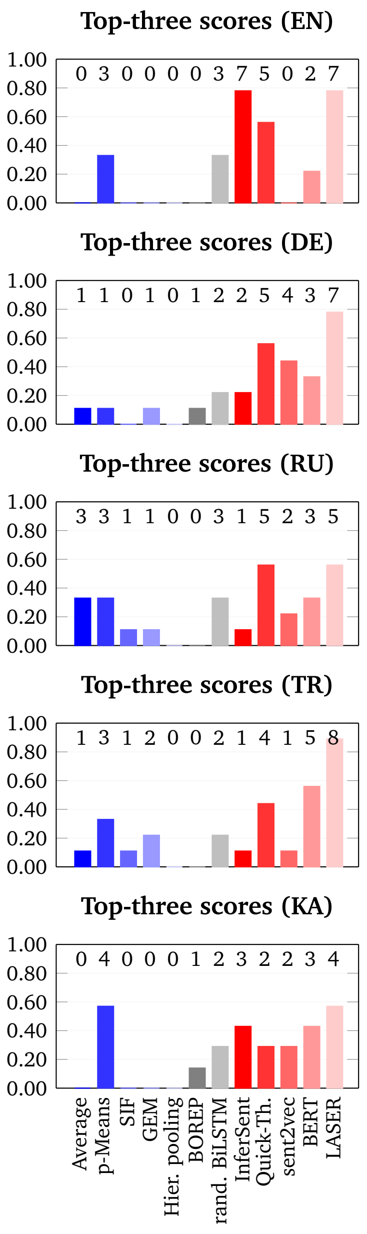


Figure 19: Top-three performances for all embeddings and languages. The bar plots show scores normalized by the number of tasks.

Furthermore, *BERT* embeddings show mediocre performance, at least in the way they are used in this thesis. Figure 19 indicates that these embeddings seem to work better in lower-resource languages, especially in Turkish (5× top-three performance). The reason for the worse top-count performance in English is that other embeddings, mainly *LASER*, *Quick-Thought* and *InferSent*, show much stronger performance due to more training data. One further aspect contributing to the worse scores is that the embeddings were **not fine-tuned on the specific tasks which is commonly done for *BERT***. Furthermore, instead of the unusual way of using the activations of the last transformer layer, it may be worthwhile to use *SBERT* (Reimers and Gurevych, 2019) or to adopt an approach similar to (Krasnowska-Kieraś and Wróblewska, 2019) who use average *BERT* embeddings. In this setup the representations are primarily seen as word embeddings which are subsequently pooled to obtain a sentence representation.

The overall results suggest that trained encoders can still be advantageous if resources are scarce. But it may sometimes be more beneficial to rely on averaging methods rather than on trained embeddings in such a case.

On the performance of compositional methods. Not only in lower-resource languages should the capabilities of averaging methods be underestimated. Instead, they should be considered competitive in general. It can be observed that some of the **averaging methods (especially *vanilla average* and *concatenated power means* embeddings) often demonstrate surprisingly good performance on a wide range of tasks.** This has already been reported by several authors (Arora et al., 2017; Wieting et al., 2016b, inter alia). Figure 19 shows that they may not be among the top three as often as trained encoder architectures, but the absolute performance gap is often small. Of course, such embeddings perform randomly on tasks involving the order of words, but they excel for example in the SubjNum task (cf. results for German, Russian and Turkish). (Conneau et al., 2018a) justify this result with *'redundancies in natural language'*, i. e. that information about subject number is not only encoded in the subject itself, but also in other words that refer to it. For example, the form of a verb is adjusted to match the number of the subject. This means that the number information is encoded in multiple locations within a sentence. (Conneau et al., 2018a) further elaborate that word embeddings are known to retain information about tense and number (Mikolov et al., 2013b), which is why a vector obtained by averaging these word embeddings also contains the respective information. **An additional advantage is that averages can be computed fast and efficiently.**

On average, the choice of a specific word embedding as a basis for averaging models does not influence the results fundamentally. However, the results deviate substantially for the WC task. For all languages, except for German, the results suggest that, above all, the *concatenated power means* embeddings exhibit much stronger performance if they are based on either *word2vec* or *Attract-Repel* word embeddings (cf. tables in appendix B on page 72). Consider for example the Russian language, where the *power means* embeddings achieve ≈ 0.99 F1 score on WC when using *word2vec* word embeddings in contrast to ≈ 0.67 F1 score when using *FastText* vectors. For English, a similar effect can be observed: On the same task the *power means* representations achieve ≈ 0.95 F1 score with *Attract-Repel* embeddings and only ≈ 0.66 F1 score with *FastText* embeddings. The data creation process for WC made it necessary to increasingly resort to Wikipedia sentences. The *word2vec* embeddings used in the context of this thesis were also trained on Wikipedia sentences which leads to less OOV words. Normally, *FastText* can deal with words it has not seen during training by averaging *n*-gram embeddings. Unfortunately, **pre-computed word vectors were employed** without using this advantage. Therefore, the performance of *FastText* is affected by more OOV words. The increased performance of *Attract-Repel* must be credited to the fine-tuning of the initial vectors on linguistic constraints.

On the universality of embeddings. Many sentence embedding techniques claim to be universal, where universality refers to the property of language representations to show strong performance, independently of the task to be solved. The design of such an embedding algorithm is a noble goal but unfortunately, **it seems that such sentence representations are not yet available**. (Perone et al., 2018) come to the same overall conclusion. The results indicate that there is no embedding that consistently shows strong performance in all probing tasks, but ***LASER* can be considered closest to a universal embedding**. Nevertheless, these embeddings do not seem to capture information about subject-verb agreement well in the English language. The general lack of universality entails that there are different embeddings scoring highly on different probing tasks. Consider as an example the results generated for *sent2vec*: This encoder seems to work extremely well for the detection of subject number in the German language (in this task it is the best-performing embedding with 0.906 F1 score), but the exact same algorithm performs very poorly when predicting the length of a given sentence (Δ to best-performing embedding $\approx$ 0.337 F1 score). Please note, that the ordering of the embeddings does not necessarily have to be consistent across all languages. Consider again the SubjNum task: For German, the *sent2vec* embedding performs best, while it is *LASER* for the English language.

On the performance of random encoders. **Random encoders sometimes show surprisingly strong performance**. The *random BiLSTM* frequently surpasses average embeddings or even trained sentence representations: In English, *vanilla average* embeddings (based on *FastText*) are outperformed 8× and the *random BiLSTM* achieves the same performance as *InferSent* on the SubjNum task. Also consider the results reported for the Voice task in Russian: In this setting the *random BiLSTM* shows the best performance among all embeddings with a relative performance improvement of approximately Δ 0.006 F1 score compared to the *concatenated power means* method based on *FastText* word embeddings. A similar effect can be observed for the Georgian language. In Turkish, the same random encoder is at position three for the Voice task. (Wieting and Kiela, 2019) already pointed out that the gap between random encoders and their trained counterparts is often smaller than expected. What is more, experiments conducted by Wieting and Kiela further demonstrate that the performance is quite stable and improves with growing embedding dimensionality. (Conneau et al., 2018a) believe that a random encoder still produces a kind of **positional encoding** which can help in certain task settings.

On the dimensionality of the embeddings. In general, one has to keep in mind that **the dimensionality of the resulting embeddings is different for almost all sentence encoding techniques**. While the majority of compositional embeddings have less dimensions (here: 300),[52] trained encoder architectures are usually high-dimensional (e. g. *InferSent* with 4,096 dimensions). Surprisingly, we found that *vanilla average* sentence embeddings occasionally perform better than the *power means* approach (1× for Georgian, 2× for English, German and Turkish and 3× for Russian, where mostly the tasks WO and SubjNum are involved). Figure 20 on the following page generally reveals, that high dimensionality is no guarantee for good probing task performance: On the WC task, the Pearson correlation between embedding dimensionality and performance is consistently negative across all languages. **This implies that increasing the dimensionality can sometimes hurt the performance, at least the performance as measured by the F1 score.**

Language-specific observations. As already mentioned above, the relative ranking of trained encoders changes in lower-resource languages. The purpose of table 18 is to provide more information about the rank stability of the embeddings in the individual probing tasks across different languages. To this end, we computed the Spearman correlations between the probing task results in English and the remaining languages for each probing task:

	Spearman correlations								
	SentLen	WC	BiShift	SVAgree	SVDist	Voice	WO	EOS	SubjNum
EN-DE	0.98	0.02	0.86	0.79	0.88	0.34	0.88	0.84	-0.05
EN-RU	0.96	0.41	0.94	0.34	0.89	0.19	0.98	0.83	0.42
EN-TR	0.89	0.41	0.90	0.12	0.87	-0.06	0.97	0.88	0.54
EN-KA	0.93	0.04	0.73	0.54	-	0.40	0.96	0.89	-

Table 18: Spearman correlations between probing task performances in different languages.

[52] An exception is the *p-means* technique whose dimensionality depends on the parameters. In this context it has a dimension of 1,500.

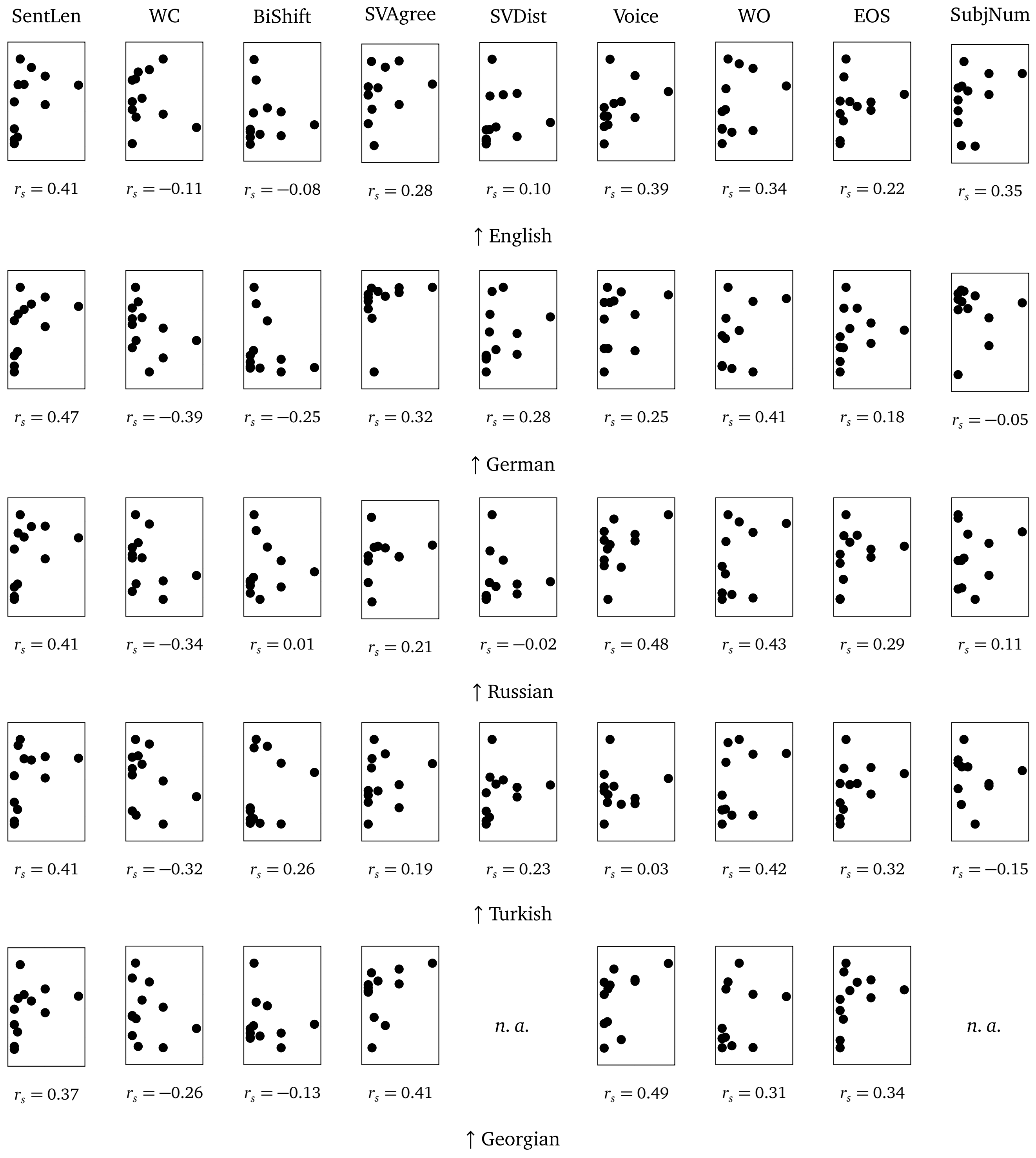


Figure 20: Pearson correlations of embedding performance (F1 scores) and embedding dimensionality. The embedding size is plotted on the x-axis, the performance on the y-axis. The correlations imply that high dimensionality is not always a guarantee for good probing task performance.

For two random variables, X and Y, the Spearman correlation is computed as follows:

$$r_s = 1 - \frac{6 \cdot \sum_{i=1}^{n}(rank(x_i) - rank(y_i))^2}{n \cdot (n^2 - 1)} \qquad r_s \in [-1, 1] \tag{8}$$

where n is the number of observations (here: the number of embeddings) and $rank(\bullet)$ outputs the rank of $\bullet$ among all observed values. The range of values a correlation coefficient can take is between -1 and 1. A positive correlation indicates that the two variables tend to increase or decrease together,[53] while negative values imply, that if one variable increases, the other one decreases and vice versa.[54] If the coefficient is equal to 0, then there is no discernible (linear) correlation between the two variables (Fahrmeir et al., 2011, pp. 135–148). The ordering of the embeddings in the tasks SentLen, BiShift, SVDist, WO and EOS is rather consistent for the other languages compared to English (indicated by strong positive correlations). In contrast to this, the ordering for WC, SVAgree, Voice and SubjNum is more influenced by the underlying language (slightly negative/low positive correlations).

Comparison of the results with the literature. The following paragraphs provide a comparison of the probing task results described above with those reported in the literature. The detection of possible discrepancies will stimulate further research and experiments in order to investigate their origin (cf. section 7.5 on page 60). One has to keep in mind that we created new data sets for the probing tasks, which means that the results are not directly comparable. Nevertheless, the linguistic properties which are probed remain the same. For this reason, only strongly diverging results will be discussed and the explanations focus more on different task setups.

❶ **Paper: Empirical Linguistic Study of Sentence Embeddings.** The paper which is most similar to our work is the publication by (Krasnowska-Kieraś and Wróblewska, 2019) who probe English and Polish sentence representations. They base their experiments on the tasks introduced by (Conneau et al., 2018a), but modified some of them. The authors include many embedding techniques into their evaluation which are not used in this thesis. These include e. g. *COMBO* (Rybak and Wróblewska, 2018), *Universal Sentence Embedding (USE)* (Cer et al., 2018) or average *BERT* embeddings. The following explanations are therefore restricted to the intersecting set of embeddings (English language):

The *vanilla average* representations based on *FastText* perform significantly worse on SentLen in our experiments (30.22 % accuracy) than reported in the literature (72.27 % accuracy). However, (Krasnowska-Kieraś and Wróblewska, 2019) use a different task setup: **Their data set comprises 90k instances**, whereas we only included 10k instances at maximum due to difficulties in acquiring enough training data in lower-resource languages and we wanted the English data to be comparable. Furthermore, the authors constructed **balanced data sets**. In contrast to this, table 12 on page 38 shows that the SentLen data set follows a natural distribution in our work. The performances on the WC task deviate in the other direction: Our experiment results in 75.92 % accuracy compared to 46.73 % accuracy reported by the authors. This can be explained by the **unequal number of classes** in the two settings: (Krasnowska-Kieraś and Wróblewska, 2019) framed the task as a 750-way classification task, whereas there are only 30 classes considered in the present work. Classification problems usually become easier with less classes to choose from. The authors found that *sent2vec* performs strongly on the WC task. There, it shows the strongest performance among all embeddings the authors considered. We could find a similar pattern: In German and Russian, it achieves approximately 0.97 F1 score. However, it achieves rather poor scores on most of the remaining probing tasks (except for SVAgree).

❷ **Paper: What you can cram into a single $&!#* Vector.** This seminal paper by (Conneau et al., 2018a) introduces to concept of probing tasks on a large scale. The authors were the first to systematically probe sentence embeddings for different linguistic aspects using a suite of ten probing tasks. Again, the largest differences can be observed for the SentLen task as well as the WC task. These differences are mainly observed for the *vanilla average* embeddings. The performance of *InferSent*, on the other hand, is comparable. Also Conneau and colleagues use more training data. They use the *SentEval* framework which includes **balanced data sets of size 120k** (100k `train`, 10k `dev`, 10k `test`). **However, the number of training examples cannot be the only difference:** A comparison of the WC results reported by (Conneau et al., 2018a) with those by (Krasnowska-Kieraś and Wróblewska, 2019) still reveals a large gap: *Vanilla average* embeddings achieve 46.73 % accuracy (Krasnowska-Kieraś and Wróblewska, 2019) while (Conneau et al., 2018a) report a score of 91.09 % (even though Conneau and colleagues consider 1k classes). Please note, that both authors use similar-sized training data sets. The paper by Conneau et al. reveals that they **used a logistic regression classifier**

[53] $r = +1 \equiv$ perfect positive linear correlation.

[54] $r = -1 \equiv$ perfect negative linear correlation

instead of an MLP, since this produced better results for them. In contrast to this, we and (Krasnowska-Kieraś and Wróblewska, 2019) employ an MLP. Furthermore, (Conneau et al., 2018a) state that they **performed hyper-parameter optimization** for the following hyper-parameters: `# hidden units` ∈ {50, 100, 200} and `dropout` ∈ {0, 0.1, 0.2}). This tuning was performed on the dedicated `dev` sets of the probing tasks. In contrast, we did not conduct such a hyper-parameter tuning.

❸ **Paper:** **Evaluation of Sentence Embeddings in Downstream and linguistic Probing Tasks.** (Perone et al., 2018) conduct experiments which are very similar to those by (Conneau et al., 2018a). They criticized that some state-of-the-art sentence encoders were not included in the evaluations conducted by Conneau and colleagues. The authors could show that an average of deep contextualized word embeddings is able to beat more sophisticated models trained on NLI data like *InferSent*. The authors also make use of the *SentEval* framework and adopted all of the probing tasks. In general, (Perone et al., 2018) report rather strong results on some of the probing tasks. Especially on WC, Perone and colleagues get surprising outcomes: The *concatenated power means* embeddings achieve 98.85 % accuracy on the test set, although the data set comprises 1k classes. Next to the reasons already mentioned, **a higher-dimensional sentence embedding must be held responsible**: Perone et al. use the `tf-hub` module[55] which is officially provided by (Rücklé et al., 2018). This model concatenates various power-means computed in several embedding spaces. We used one embedding space only (*FastText*). Also, the authors followed the approach of (Conneau et al., 2018a) and **used a logistic regression classifier** in this task.

Furthermore, (Perone et al., 2018) state that they evaluated the sentence embeddings by **leveraging a nested cross-validation** approach for some of the downstream tasks (used for hyper-parameter optimization in the context of x-validation). However, the paper leaves it open whether they proceeded analogously for the probing tasks and also the paper does not give any indication with respect to which hyper-parameters were optimized. **We attempted to get in touch with the authors on this, but our e-mail remained unanswered.**

Finally, table 19 summarizes the main differences between the task setup in this work and the literature:

Literature	This work
• Monolingual or bilingual evaluation	• 5 languages: EN, DE, RU, TR, KA
• Larger data sets (90k/120k)	• Data sets comprise 10k instances at maximum
• Balanced data sets	• Partially imbalanced data sets
• Hyper-parameter optimization	• No hyper-parameter optimization
• Partially different classifiers	• Usage of an MLP for all tasks

Table 19: Main setup differences for the probing tasks.

❹ Probing Multilingual Sentence Representations With X-Probe. In this paper, (Ravishankar et al., 2019) evaluate multi-lingual embeddings by aligning the sentence encoding models. As already introduced in section 2 on page 3, the authors first train the weights of an encoder for the English language by exploiting NLI data. The sentence representations for the remaining languages are then aligned on parallel corpora by minimizing the mean squared error between the English sentence representations and the target-language representations. The resulting encoders are subsequently used and evaluated on the probing task suite introduced by (Conneau et al., 2018a). However, the general evaluation procedure is entirely different to the approach used in this thesis. E. g. there are no parallel corpora in the context of the present work and also the set of encoding architectures is completely disjoint to the ones considered in this thesis. **Therefore, a direct comparison of the results does not make sense and is therefore omitted.**

Summary

The main setup differences constitute the data set size as well as the class balance. Furthermore, many authors perform hyper-parameter optimization for the classifier architectures which was not done in this context. Also, different classifiers (e. g. logistic regression) are sometimes used in the literature. Section 7.5 on page 60 tries to isolate the effects of these setup choices.

[55] `https://github.com/UKPLab/arxiv2018-xling-sentence-embeddings` (retrieved: September 09, 2019)

Similarly to the probing tasks in the previous section, figure 22 on the next page illustrates the main results obtained for the downstream applications across all languages. The following paragraphs provide a description of the main observations and also compare them to results in the literature. The detailed results for the downstream tasks can be found in the appendix (cf. appendix C on page 79).

First of all, it is immediately apparent that **the results in downstream applications across different embedding methods are not as volatile as compared to probing task performances** (cf. figure 18 on page 47): Simple compositional models are most of the time just behind more sophisticated methods like *InferSent* or *LASER* and sometimes even outperform trained encoders in low-resource languages. This narrow gap in performance suggests that the choice of a particular embedding is not as decisive in more realistic downstream applications, but it is still not completely unimportant. (Perone et al., 2018) depict similar bar plots (cf. figure 21, abridged) and this trend is also clearly discernible in their results (top row: probing tasks where performance is partially very different, bottom row: downstream tasks where results are more stable across the embeddings). It is not quite clear why the patterns in these two types of tasks are so different. **We assume that the artificial nature of probing tasks is responsible for the deviations.**

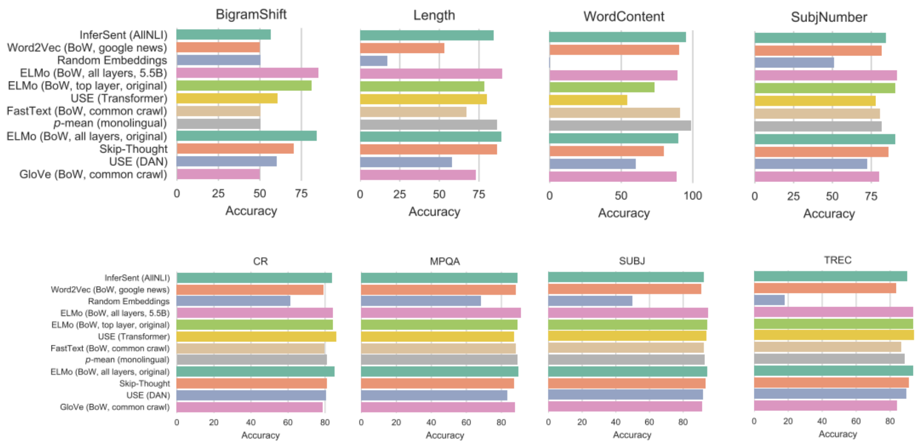


Figure 21: Results for selected probing and downstream tasks as reported by (Perone et al., 2018).

Secondly, it is surprising that **random encoders score very highly in all languages**. The results we obtained show that such embeddings are able to outperform average embeddings (cf. TREC task in English). (Wieting and Kiela, 2019) find a similar pattern and therefore conclude that **random encoders *'[constitute] a much stronger baseline'* than compositional models**. Comparing these results to (Perone et al., 2018) (cf. figure 21), one notices that they report the performance of random encoders to be substantially worse compared to the rest of the embeddings considered in their paper. Unfortunately, Perone and colleagues do not specify exactly how these embeddings were created. Differences can be caused by a different dimensionality or by a completely different encoding architecture. Despite the overall impressive performances of random encoders in this work, one can still identify a gap between those embeddings and trained sentence encoders in the English language. Nevertheless, these differences are much *'smaller than [one] would have hoped'* to phrase it in the words of (Wieting and Kiela, 2019).

ArgMin task (EN). For the ArgMin task in English, we see that the classifier used in this work **achieves comparable results with the literature** (or even better), although the classifiers which are learned on top of the embeddings are entirely different in the two cases. In this work, the simple MLP is used as specified in section 6.5 on page 45. Next to the actual sentence representation, this model **additionally receives as input a sentence embedding which encodes the topic of the respective sentence**. (Stab et al., 2018), on the other hand, use a more sophisticated BiLSTM network named *contextual BiLSTM* whose cells are altered in a way such that the input and cell memory gates are additionally fed with the topic information. However, (Stab et al., 2018) use a **different evaluation methodology**: They train the model on seven out of eight available topics and test the model on sentences belonging to the remaining topic which allows to draw conclusions concerning the extent to which the model is able to generalize to unseen topics. The MLP in this work is evaluated using a simple cross-validation approach, **disregarding the affiliation of arguments to specific topics**. This task is much easier, since the model was trained on sentences belonging to all topics and **therefore has advantages over the *contextual BiLSTM* from the literature**.

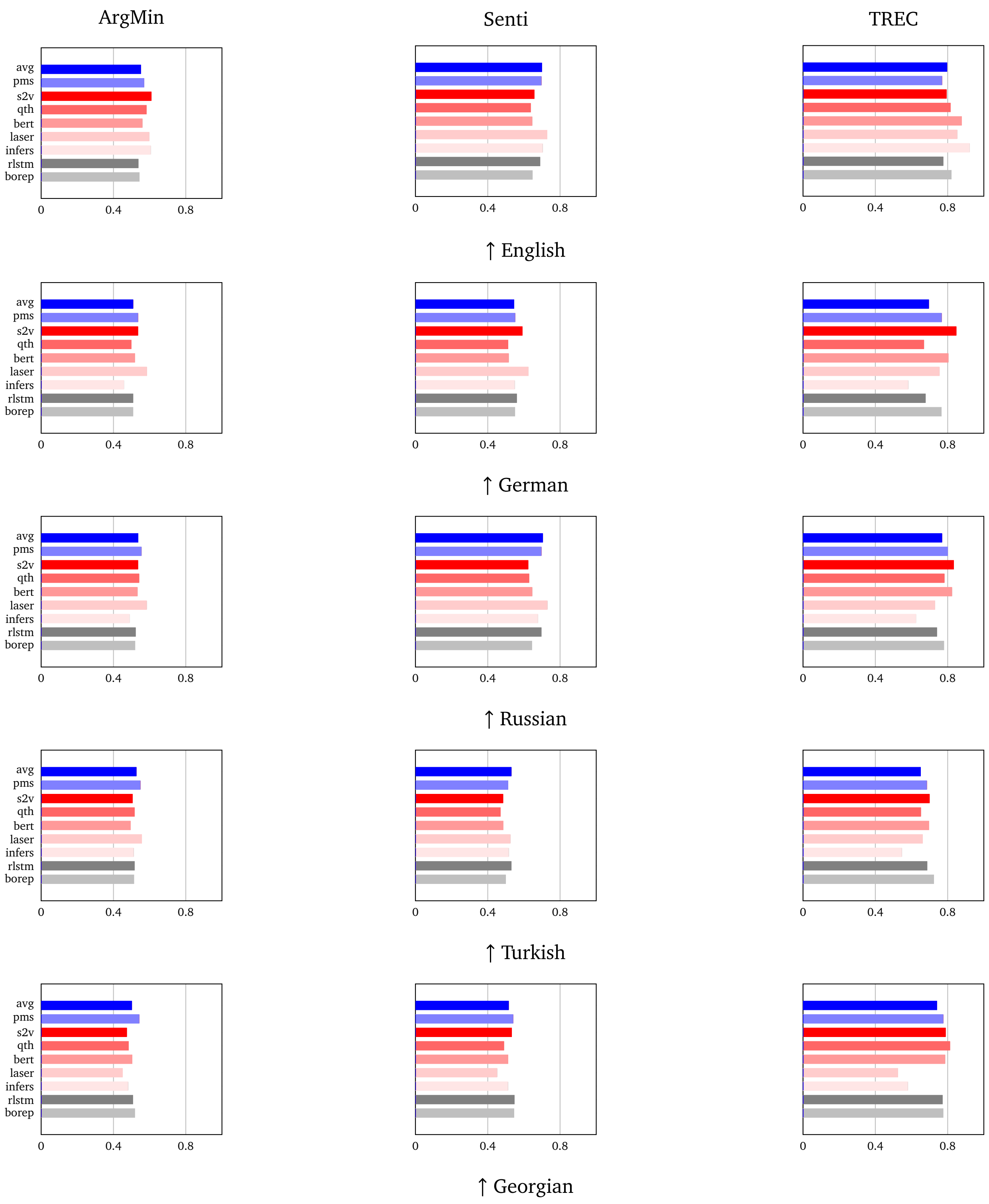


Figure 22: Downstream task results for all languages (F1 scores). For reasons of legibility only the most important embeddings are shown where 'avg' stands for *vanilla average*, 'pms' refers to the *concatenated power means* embedding, 'qth' is the abbreviation for *Quick-Thought* and finally, 'rlstm' denotes the *random BiLSTM*). Average embeddings were plotted using a blueish color (all average methods use *FastText* word embeddings) whereas trained sentence encoders are shown in a reddish color. The random BiLSTM encoder serves as a baseline and was printed in gray color. In contrast to the probing tasks, the data sets for all languages are similar in size (most of the time identical).

TREC task (EN). As table 20 demonstrates, the English results for this task deviate in part from the literature. Especially the *power means* embedding performs much worse in our evaluation (Δ -9.20 pp accuracy). One reason for this is, that (Rücklé et al., 2018) concatenated mean embeddings obtained from four different embedding spaces, namely *Attract-Repel*, *GloVe*, *word2vec* and *MorphSpecialized* (Vulic et al., 2017), whereas only *FastText* embeddings are used in this work (without computing vectors for OOV words). Furthermore, the authors normalized the embeddings. (Rücklé et al., 2018) also report results for single embedding spaces: For *GloVe* they obtain 85.60 % accuracy which is much closer to our results (Δ -3.80 pp accuracy).

Furthermore, *Quick-Thought* exhibits scores which are much worse than reported in the literature (Δ -11.00 pp accuracy without hp-tuning). This circumstance is due to the fact that the original English model was trained on book corpora and *GloVe* embeddings, whereas our model uses *word2vec* embeddings as a basis and was furthermore trained on Wikipedia. **The different styles of writing (encyclopedic vs. novel-like) in these two corpora may cause differences in performance**. *InferSent* embeddings, on the other hand, show improved performance in our experiments (Δ +0.60 pp accuracy without hp-tuning) compared to the literature. Despite worse results on probing tasks, our *BERT* embeddings achieve better performance (91.00 % accuracy) than *SBERT* with 89.6 % and 87.4 % accuracy for the *base* and *large* versions, respectively (Reimers and Gurevych, 2019).

The partially worse results still require further investigation. They do not change substantially, even with hyper-parameter tuning enabled (`dropout` $\in \{0, 0.1, 0.2\}$, `# hidden units` $\in \{50, 100, 200\}$, early-stopping with patience set to 5): The embeddings achieve slightly better performances, without major influences on their relative ordering. Also, results in the literature are often produced **using a logistic regression classifier** with different activation functions. Despite these differences, the general trends reported for the English TREC task can still be observed in our results: Random encoders tend to outperform averaging methods, while there is still a gap to trained encoders.

Embedding	**Accuracy** (literature)	**Accuracy** (this work)	**Relevant Paper**
p-Means	91.00 / 85.60 (one emb. space)	81.80 (81.80)	(Rücklé et al., 2018)
sent2vec	83.80	78.20 (79.40)	(Pagliardini et al., 2018)
Quick-Thought	92.40 (diff. model)	81.40 (83.00)	(Logeswaran and Lee, 2018)
InferSent	90.80	91.40 (92.20)	(Conneau et al., 2017)
(S)BERT	89.60 (diff. model)	90.80 (90.80)	(Reimers and Gurevych, 2019)
BOREP	88.80	84.50 (84.60)	(Wieting and Kiela, 2019)
Random LSTM	86.50	79.60 (83.80)	(Wieting and Kiela, 2019)

Table 20: Important accuracy scores on the TREC task reported in the literature. The results in parantheses are with hyper-parameter optimization.

SENTI task. It is difficult to compare the results for the SENTI task directly with the literature, since these data sets are not included in the standard set of downstream tasks considered in the *SentEval* framework or were created from scratch in the context of this thesis (e. g. the Georgian SENTI data set). We only found results by (Lee et al., 2017) for the *GermEval* data. Unlike us, the authors used a stacked classifier and trained the models (*sent2vec* and *SIF*) on different corpora (including the GermEval data). Also, the embedding dimensionality is different, which is why a comparison is omitted.

Effect of different word embeddings. On average, compositional models on downstream tasks work best with *FastText* embeddings as a basis, whereas *Attract-Repel* performs worst (cf. appendix C on page 79 for detailed results).

Low-resource languages. **The gap between compositional models and trained encoders vanishes when the language belongs to the low-resource category. Furthermore, random encoders perform convincingly**. For Turkish and Georgian, we notice from figure 22 on the previous page that, most of the time, either the *power-means* embedding or the *random encoder* is superior. The bad performance of trained sentence encoder architectures is again a direct consequence of the low-resource property of the Georgian language. Especially the *LASER* embeddings, whose performance is superior in the other languages, scores comparably poorly on Georgian data sets. But also *InferSent* exhibits a drop in performance, since the models for German, Russian, Turkish and Georgian were trained on only a small portion of the *AllNLI* data set.

Summary

Diverging results on the TREC task are mainly credited to different embedding models (e. g. *p-Means*, *Quick-Thought*) as well as missing hyper-parameter optimization. Nevertheless, the main trends reported by Wieting et al. can still be observed in the results for the English TREC task: Random encoders tend to outperform averaging methods, while there is still a gap to trained encoders. Furthermore, compositional embeddings and random encoders convince with strong performance in low-resource languages.

7.4 Correlation Analysis of Probing Task and Downstream Task Performances

The following analysis aims to investigate the extent to which probing tasks are predictive for downstream application performance in the target languages. One method to examine this interrelationship is to compute the correlations between the respective results. We used Spearman correlations in our experiments in order to align our work with the literature. In the following, consider a probing task $\mathfrak{P}$ and a downstream task $\mathfrak{D}$: A strong positive correlation between the two tasks indicates that all embeddings which perform well on $\mathfrak{P}$ also achieve high performances on $\mathfrak{D}$ and vice versa. Negative correlations, on the other hand, reveal that embeddings scoring highly on $\mathfrak{P}$ perform poorly on $\mathfrak{D}$ and vice versa. In the present context this can be translated as follows: A positive correlation between $\mathfrak{P}$ and $\mathfrak{D}$ suggests that the linguistic property probed by $\mathfrak{P}$ is necessary to excel at $\mathfrak{D}$. A negative correlation points to the fact that the knowledge about this linguistic feature is inversely related. Thus, an embedding should not retain this information in such a case.

Correlations in the English language. Figure 23 plots the Spearman correlations obtained for the English language (F1 scores). A **reddish** color denotes positive correlations, a **blueish** color refers to negative correlations. Results amounting to an absolute value of less than 0.2 were set to 0.00. Furthermore, only the average embeddings based on *FastText* are considered in the following analysis.

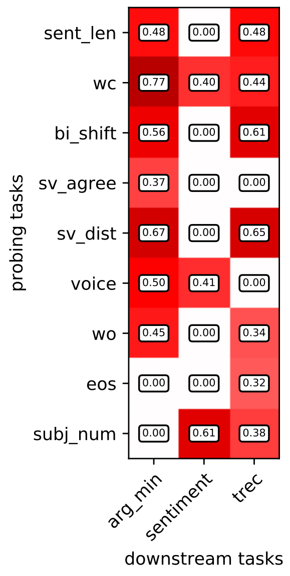


Figure 23: Spearman correlations between probing and downstream task performances in English **measured by F1 score**. Reddish values correspond to positive, blueish values to negative correlations.

First of all, we notice that the correlations we obtained for the SentLen task are rather high for TREC and ArgMin. This contradicts the finding reported by (Conneau et al., 2018a) who found that the SentLen task is mainly negatively correlated with almost all downstream tasks (cf. figure 24 on the next page, left image). However, (Eger et al., 2019) have retrospectively correlated the results published by (Perone et al., 2018) and **found the exact opposite of what Conneau and colleagues report**. In this analysis, the SentLen task shows the strongest connections to all downstream tasks on average (cf. figure 24 on the following page, right image). (Eger et al., 2019) conjecture that different sets of sentence embedding algorithms considered in the two papers are responsible for the divergence of the results. Conneau and colleagues use a set of related embeddings, whereas Perone et al. consider a more diverse set. **Different algorithms might focus on different linguistic aspects which influences the correlations.**

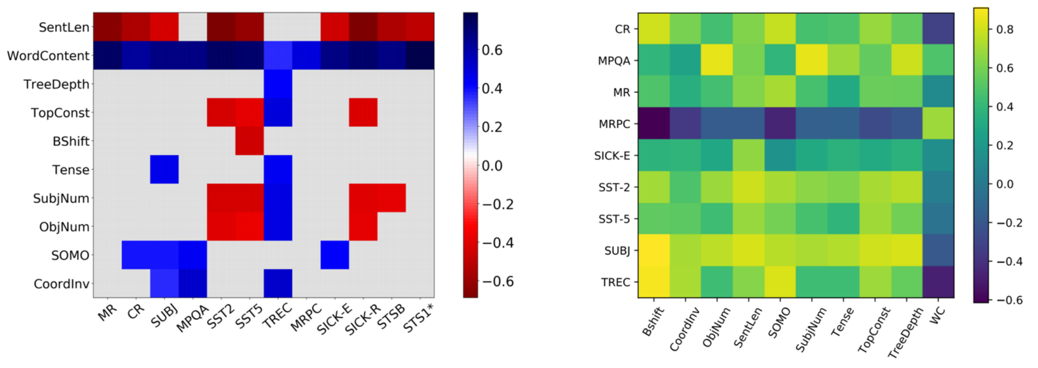


Figure 24: Correlations between probing and downstream task performances from the literature. Left: Correlations reported by (Conneau et al., 2018a). Please note that the color scheme is inverted.; Right: Correlations of the results by (Perone et al., 2018). The correlations were computed retrospectively by (Eger et al., 2019).

Task	FastText$_{max}$	FastText$_{mean}$	BERT$_{max}$	BERT$_{mean}$	COMBO$_{max}$	COMBO$_{mean}$	sent2vec$_{ns}$	sent2vec$_{orig}$	LASER	USE	r_s ENTAIL
SENTLEN	52.55	72.27	72.66	82.13	85.03	87.38	71.56	64.76	85.98	60.00	**-0.26**
WC	24.44	46.73	35.24	45.53	9.39	11.05	59.96	79.23	59.79	43.11	**0.72**
VOICE	84.13	89.47	89.77	92.40	98.48	98.41	88.73	89.04	92.85	86.61	**-0.36**
SUBJNUM	73.87	81.43	88.43	90.75	93.19	93.37	82.27	80.88	94.21	81.65	**-0.02**
ENTAILMENT	76.72	76.86	77.71	77.11	72.82	72.58	78.59	78.26	83.26	81.77	–

Table 21: Results reported by (Krasnowska-Kieraś and Wróblewska, 2019) for several probing tasks as well as ENTAILMENT.

Conneau and colleagues furthermore highlight the very strong positive correlations of the WC task with all downstream applications. This finding can be replicated in our experiment for the English language. This suggests that storing lexical information in the embeddings helps in downstream tasks. Consider for example the SENTI task, where the goal is to classify a sentence (or a snippet of text) into either positive or negative sentiment: If a sentence predominantly contains negatively (positively) connoted words, it is much more likely that this sentence also describes a negative (positive) sentiment. The ARGMIN task may serve as an additional example: The data set consists of sentences arguing either for or against a specific topic. If an embedding encodes information about argumentative words like e. g. *'on the other hand'*, *'likewise'*, *'despite'*, *'nevertheless'*, *'as a consequence'*, *etc.*, the classification of sentences into either supportive or confuting is accomplished much better. Finally, also the TREC task profits from this knowledge: Storing information about the interrogative pronoun (question words like e. g. *'what'*, *'where'* and *'how'*, etc.) directly gives hints as to what type of answer is expected. The results by (Perone et al., 2018), on the other hand, indicate that knowledge about word containment is not as important for good performance in downstream applications **which sounds counter-intuitive**.

The correlations calculated by (Conneau et al., 2018a) furthermore indicate that detecting legal word orders is most of the time not indicative for good or bad performance in downstream applications. **Our results suggest a different conclusion:** The ARGMIN task as well as the TREC task exhibit high correlations to the BISHIFT probing task. This is compliant with the results by Perone et al. for which (Eger et al., 2019) compute a strong positive correlation of TREC and BISHIFT. We found that the closely related WO task which also involves word order behaves analogously to BISHIFT. (Conneau et al., 2018a) further point out that **the TREC task has strong positive correlations with almost all probing tasks**. This finding can be reproduced in the context of our experiments. Only the tasks SVAGREE and VOICE are not connected to question type classification. Unfortunately, the SVAGREE task is not contained in the set of tasks introduced by Conneau et al., which is why a comparable figure is missing here.

The VOICE task has positive correlations with the downstream tasks ARGMIN and SENTI. If the accuracy metric is used (cf. figure 25 on the next page), the correlations become even stronger and VOICE is additionally correlated with TREC. This suggests that voice information is helpful when it comes to performing well on downstream tasks. This finding is not intuitively obvious and the reasons for the importance of retaining voice information in sentence embeddings require further investigation. Unfortunately, this task is not included in the set of probing tasks shipped with *SentEval*, but (Krasnowska-Kieraś and Wróblewska, 2019) probed sentence embeddings using this task. The authors do not report probing task-downstream task correlations, but they report the accuracy values. Hence, it is possible to calculate the correlations retrospectively. The relevant results from their paper are summarized in table 21 on the preceding page. As can be seen from the table, the correlation between the probing task Voice and the downstream task Entailment is **negative**. **This means that Voice is not always as predictive of downstream task performance as implied by the results we obtained.**

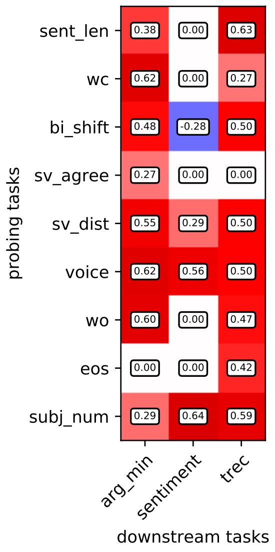


Figure 25: Spearman correlations between probing and downstream task performances in English **measured by accuracy**. Reddish values correspond to positive, blueish values to negative correlations.

Summary

Many findings from the literature can be reproduced, yet differences can be observed due to different embeddings considered, different data sets or different probing and downstream tasks.

Correlations in other languages. The following paragraphs discuss the results for the remaining target languages, namely German, Russian, Turkish and Georgian. The figures 26 on the next page, 27 on the following page, 28 on the next page and 29 on the following page show the respective results.

The results for the other languages behave differently when compared to the English language. In comparison to the English results, we notice first of all that a larger number of correlations is below an absolute value of 0.20, which is why they were set to 0.00. **What is more, many correlations have swapped into the negative range.** The strongest negative correlation can be observed for the Georgian WO task, for which we computed a correlation with Senti amounting to -0.73. Also the related BiShift task shows a slight negative correlation with this downstream task in Georgian. The main reason for this may be the fact that it is possible to arrange words rather flexibly in the Georgian language (cf. section 5.3.3 on page 25), which renders the encoding of such information futile. Furthermore, an embedding has a finite number of dimensions. Encoding word order information might prevent the encoding of linguistic properties which are more relevant in the downstream tasks and might thus harm performance. While the TREC task was reported to be correlated considerably positively with all probing tasks in the English language, **we could not reproduce this result in any of the other target languages**. The correlations for this task are either close to zero or negative. Furthermore, there are extremely strong positive correlations for the WC task in German and less strong, but still noticeably positive correlations in Russian and Georgian. For the Turkish language we cannot confirm this finding: **There, the knowledge about word containment does not seem to be particularly decisive for downstream task performance**. This is surprising in light of the explanations given for the English language. In Turkish, the correlation between WC and Senti is even negative. This implies that either the explanations presented above are wrong, or they are at least not generalizable to all languages. Overall, the correlations in the Turkish language are less strong (except for the Senti task with Voice and SubjNum). It is not entirely clear why this is the case for Turkish.

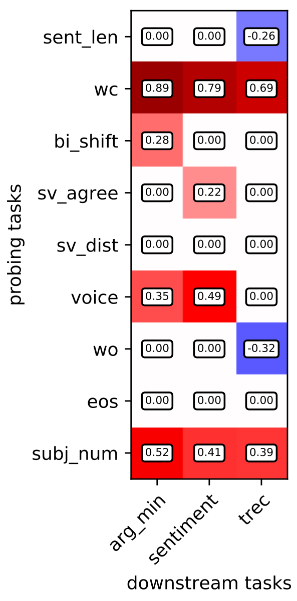


Figure 26: Spearman correlations in German.

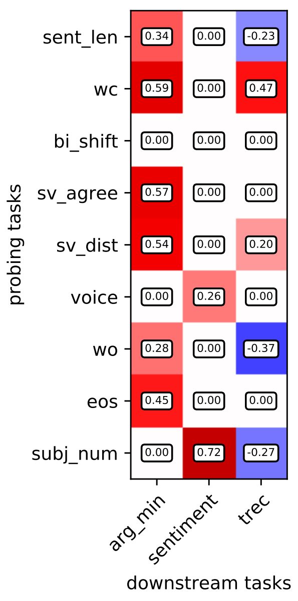


Figure 27: Spearman correlations in Russian.

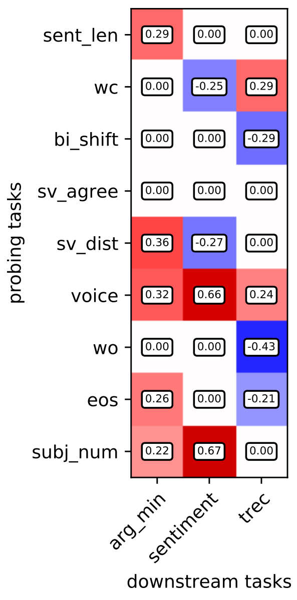


Figure 28: Spearman correlations in Turkish.

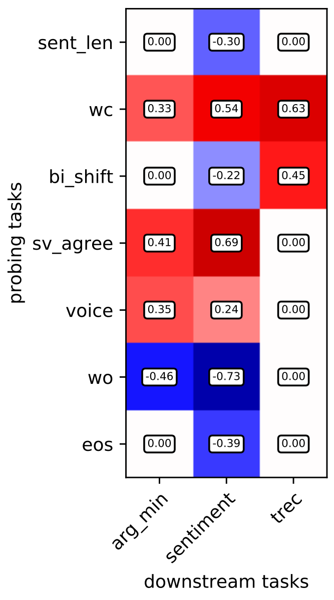


Figure 29: Spearman correlations in Georgian.

Summary

The results for the remaining languages are different compared to English. Many correlations have fallen below an absolute value of 0.20 or have swapped into the negative range entirely. The strongest negative correlation can be observed for the Georgian language. Also the correlations of the TREC task with the set of probing tasks vanish. Turkish is the only language for which the knowledge about word containment is not predictive for downstream task performance.

One common problem in the evaluation of sentence embeddings on probing tasks is that different evaluation settings are used quite frequently. These differences are manifold: First of all, the size of the data sets are not standardized and vary as a consequence. Secondly, the research community did not agree on a single classifier type which is used to test the embeddings. Sometimes neural networks with a hidden layer are used, but also logistic regression is quite common. These two approaches differ in their representational power, thus influencing the final results. Furthermore, we detected settings where hyper-parameters like the number of hidden units or the dropout rate of an MLP are tuned (Conneau et al., 2018a), others use a fixed model architecture for the evaluation and do not mention hyper-parameter tuning explicitly (Perone et al., 2018). I. e. it remains unclear whether or not they performed hyper-parameter optimization; and even if they did, they did not specify which hyper-parameters were tuned. Finally, the balance of the classes may have an influence on the performance of individual embeddings. On the one hand, (Shi et al., 2016) used an imbalanced VOICE data set for the evaluation (majority class amounts to 82.8 % for active voice), whereas, on the other hand, (Conneau et al., 2018a) created balanced data sets within the *SentEval* framework. The data sets in this work are partially imbalanced (cf. tables 12 on page 38, 13 on page 39 and 14 on page 39), where especially the tasks SENTLEN and WC are concerned.

These different setups motivated a detailed *'stability analysis'* whose goal is ❶ to investigate the reasons for different results in this work and the literature, and ❷ to see which of the above mentioned factors have an influence on the ordering of the embeddings. A review of the literature revealed that such a comprehensive study has not been conducted before. The following lines describe the experimental setup in greater depth: The analysis is performed for English and partially for Georgian using the example of the WC task. The idea is to systematically change the factors of the evaluation setting one at a time (*ceteris paribus*), such that the individual effects can be recorded. For reasons of time, it was not possible to perform this analysis exhaustively for all factor combinations. Instead, some combinations were picked and analyzed. The factors are as follows: `size` of the data set, `classifier`, `class balance` and `hyper-parameter tuning`.

Factors `size` and `classifier`. The first two factors which we investigate are `size` and `classifier`: We began by creating data sets of varying sizes for the WC task. Since data sets which are too large require an excessive amount of computational resources, we decided to limit the size to a maximum of 60k instances. Four different data sets were created comprising 5k, 10k, 30k and 60k instances, respectively. The data sets were generated, such that data sets of smaller sizes are real subsets of larger data sets. **This entails that the 10k WC task data set is not identical to the one used in earlier experiments. Hence, minor deviations in performance must be expected as a consequence.** The investigation of `classifier` effects requires that a set of diverse classification algorithms be chosen. To this end, we browsed the *scikit-learn* library for adequate classifiers. The list below shows the classifiers which were used and also gives information with respect to the hyper-parameters used in the experiments:

Hyper-parameter	Argument	Explanation
❶ Logistic regression (LR)		
`C`	3.75	Regularization parameter
`solver`	lbfgs	Optimization technique
`multi_class`	multinomial	Multi-class setting
❷ Random forest (RF)		
`n_estimators`	250	Number of trees in the ensemble
`criterion`	gini	Splitting heuristic
`max_depth`	70	Maximum depth of single tree
❸ Gaussian naïve bayes (NB)		
no hyper-parameters		

❹ Support vector machine (SVM)

`kernel`	linear	Kernel type
`C`	3.0	Regularization parameter

❺ Neural network without hidden layer (NN) Keras

`n_epochs`	100	Number of training epochs
`loss`	categorical cross-entropy	loss function
`optimizer`	adam	Optimization technique
`act_func`	sigmoid	Activation function

❻ Neural network with hidden layer (NN_H) Keras

`n_hidden`	50	Number of neurons in hidden layer
`drop_out`	0.00	Dropout rate

Other arguments equal to `NN`

These hyper-parameters were kept fixed for the factors `size` and `classifier` in the first half of this experiment. In order to obtain more stable results, we furthermore repeated the experiment, such that three individual results were available. The numbers listed in the following tables were produced by averaging the results over these three runs. Within each run we ensured that **all classifiers are evaluated on the same train-test splits across all embeddings** using an approach called **random subset evaluation**. Also, the random states of the classifiers were set, such that e. g. different weight initializations (for neural networks) or different attribute subset selections (for random forests) do not influence the results. **NB:** Results for the Support Vector Machine (SVM) classifier were produced only for 5k and 10k instances due to excessive computation time.

Table 22 shows the Spearman correlations obtained on the WC task by using different `classifiers`, whereas table 23 on the following page focuses on correlations between varying data set `sizes`. This evaluation serves two purposes: ❶ **Firstly, it is possible to determine a data set size for which no more considerable changes in the ranking of the embeddings take place.** ❷ **Secondly, the experiment allows for conclusions with respect to the rank stability of the embeddings across different classifiers.** We omitted detailed results here. In depth test results can be found in appendix D on page 83).

10k instances (EN)						
LR	1.00					
NN	0.91	1.00				
NN_H	0.83	0.94	1.00			
RF	0.41	0.66	0.81	1.00		
SVM	0.85	0.90	0.94	0.76	1.00	
NB	0.84	0.83	0.76	0.34	0.78	1.00
	LR	**NN**	**NN_H**	**RF**	**SVM**	**NB**

Table 22: Spearman correlations computed for different classifiers (WC task, EN, 10k instances).

First of all consider the results reported in table 22 which shows the Spearman correlations of the embedding ordering when different classifiers are employed. The numbers in the table were produced on a fixed-sized data set comprising 10k instances. When looking at the numbers, we realize immediately that **the ordering of the embeddings is in general rather unstable across different classifiers** (correlations move in the range from 0.34 to 0.94). This holds especially true for the RF classifier for which the correlations deviate drastically from the remaining classifiers. We can reproduce this finding on the 5k, 30k and 60k data sets (cf. appendix D on page 83 for more information). Consistent lower correlations can also be observed for NB. Our experiments also show that neural architectures outperform the RF classifier and NB. For *InferSent* embeddings on 60k instances we observed the following differences in performance: Δ NN_H $\longleftrightarrow$ RF/NB +0.21/+0.41 F1 score. (cf. table 24). As expected, the correlation between the LR classifier and NN is noticeably positive, since both models are closely related. Also the positive correlation between the two neural architectures (NN_H and NN) is considerable. **In summary, our results suggest that the choice of a specific classifier cannot be made blindly. Furthermore, classifiers like NB or RF should not be used, since they are consistently outperformed by neural architectures or logistic regression in our experiments.**

LR classifier (EN)				
5k	1.00			
10k	0.99	1.00		
30k	0.93	0.94	1.00	
60k	0.90	0.92	0.99	1.00
	5k	**10k**	**30k**	**60k**

NN classifier (EN)				
5k	1.00			
10k	0.98	1.00		
30k	0.90	0.96	1.00	
60k	0.88	0.94	0.99	1.00
	5k	**10k**	**30k**	**60k**

NN_H classifier (EN)				
5k	1.00			
10k	0.95	1.00		
30k	0.87	0.94	1.00	
60k	0.84	0.92	0.99	1.00
	5k	**10k**	**30k**	**60k**

RF classifier (EN)				
5k	1.00			
10k	0.99	1.00		
30k	0.97	0.97	1.00	
60k	0.97	0.97	1.00	1.00
	5k	**10k**	**30k**	**60k**

NB classifier (EN)				
5k	1.00			
10k	1.00	1.00		
30k	0.94	0.94	1.00	
60k	0.92	0.92	0.99	1.00
	5k	**10k**	**30k**	**60k**

Table 23: Spearman correlations per classifier computed for different data set sizes (WC task, EN).

Table 23 presents the rank stability with different data set sizes. We notice that for most of the classifiers, **the relative rankings between the embeddings change substantially between 5k training examples and a training set size of 60k instances**. Consider for example the numbers computed for the NN_H classifier (top right), where the Spearman correlation between 5k and 60k instances amounting to 0.84 is rather low. At the same time, the correlation between 30k and 60k training examples is very close to 1.0 (also for the remaining classifiers). This means that the additional 30k training examples do not entail any noticeable changes in the ordering of the embeddings. **We therefore conclude that a data set size of 30k instances is sufficient to obtain a stable embedding ordering (at least on the WC task in the English language).** For the RF classifier (bottom left), the correlations are already quite high between 5k and 60k training examples. **This means, that the ordering of the embeddings using this classifier is already relatively stable, even when training on small data sets.**

60k instances	LR		NN		NN_H		RF		NB	
	F1	R	F1	R	F1	R	F1	R	F1	R
Vanilla Average	0.66	6	0.78	8	0.81	6	0.31	9	0.38	7
GEM	0.81	5	0.81	5	0.78	8	0.36	8	0.48	5
SIF	0.65	7	0.79	7	0.81	7	0.38	7	0.43	6
p-Means	0.45	10	0.80	6	0.82	4	0.45	5	0.23	9
hier. pooling	0.45	9	0.51	12	0.52	12	0.24	11	0.21	10
BOREP	0.40	11	0.66	10	0.67	10	0.59	3	0.15	12
Random BiLSTM	0.36	12	0.68	9	0.75	9	0.30	10	0.16	11
sent2vec	0.83	4	0.82	4	0.81	5	0.39	6	0.64	1
Quick-Thought	0.86	3	0.92	2	0.91	3	0.57	4	0.50	4
LASER	0.88	2	0.91	3	0.92	2	0.72	2	0.55	3
BERT	0.57	8	0.59	11	0.57	11	0.22	12	0.27	8
InferSent	0.96	1	0.97	1	0.98	1	0.77	1	0.57	2

Table 24: Stability analysis results for 60k instances in (WC task, EN).

Table 24 shows detailed results of the embedding performances on the 60k data set: We see that the numbers for some of the embeddings are still not in the range reported by (Conneau et al., 2018a; Perone et al., 2018). Consider for instance the *vanilla average* embedding for which Conneau and colleagues report 91.60 % accuracy on the WC task. The result for the NN_H classifier listed above (0.81) indicates that this still outperforms our score (the table shows F1 scores, the respective accuracy score on the data set is 86.31 %), although there are less classes in our setup. **Our results further**

suggest that the neural network without hidden layer (`NN`) performs worse than the MLP (`NN_H`). (Conneau et al., 2018a; Perone et al., 2018) removed the hidden layer for the evaluation on the WC task because this gave better results in their experiments.[56] We were unable to observe this effect in our results. As table 24 on the previous page reveals, the `NN` classifier is outperformed by the `NN_H` classifier for 8 out of 12 embeddings. **The overall results suggest that the size of the data set as well as the classifier architecture are not the only effects to be made responsible for the deviations.** Before we examine the effects of the other factors (`class balance` and `hyper-parameter tuning`), we want to investigate whether the findings above can be reproduced for the Georgian language. In this context, the following results were obtained:

10k instances (KA)						
LR	1.00					
NN	0.88	1.00				
NN_H	0.89	0.98	1.00			
RF	0.45	0.49	0.62	1.00		
SVM	0.99	0.90	0.92	0.50	1.00	
NB	0.61	0.76	0.71	0.13	0.60	1.00
	LR	**NN**	**NN_H**	**RF**	**SVM**	**NB**

Table 25: Spearman correlations computed for different classifiers (WC task, KA, 10k instances).

LR classifier (KA)				
5k	1.00			
10k	0.94	1.00		
30k	0.93	0.96	1.00	
60k	0.91	0.95	0.99	1.00
	5k	**10k**	**30k**	**60k**

NN classifier (KA)				
5k	1.00			
10k	0.97	1.00		
30k	0.88	0.93	1.00	
60k	0.85	0.88	0.98	1.00
	5k	**10k**	**30k**	**60k**

NN_H classifier (KA)				
5k	1.00			
10k	0.99	1.00		
30k	0.87	0.91	1.00	
60k	0.72	0.77	0.94	1.00
	5k	**10k**	**30k**	**60k**

RF classifier (KA)				
5k	1.00			
10k	0.99	1.00		
30k	0.97	0.97	1.00	
60k	0.97	0.97	1.00	1.00
	5k	**10k**	**30k**	**60k**

NB classifier (KA)				
5k	1.00			
10k	0.94	1.00		
30k	0.95	0.98	1.00	
60k	0.97	0.99	0.99	1.00
	5k	**10k**	**30k**	**60k**

Table 26: Spearman correlations per classifier computed for different data set sizes (WC task, KA).

Overall, the analysis of the embedding orderings using different classifiers yields a similar picture for Georgian compared to the English language (cf. table 25): The `RF` classifier still exhibits the lowest correlation to all other classifiers. We observe the lowest correlation between `RF` and `NB` (0.13). Table 26 furthermore confirms that **a data set size of approximately 30k instances is sufficient** in order to obtain a stable embedding ordering. Only the correlation between the sizes 30k and 60k for the `NN_H` classifier amounts to a slightly smaller number (0.94) compared to English (cf. table 23 on the preceding page). An interesting observation is that the correlations for the `NB` classifier between 5k and 60k instances (0.97) is higher than between 5k and 10k (0.94) which means that the ordering changes when going from 5k to 10k, but approaches the original ranking again with more data.

Factors `class balance` and `hyper-parameter tuning`. The results produced for varying the factors data set `size` as well as `classifier` show that still certain differences persist between the evaluation results in this work and the literature. In the following paragraphs, we shift the focus towards the other two factors `class balance` and `hyper-parameter tuning` in the English language. As table 12 on page 38 depicts, the data set for the WC task we used previously is one among the most imbalanced ones. **For the following experiment, we therefore created a new 10k English data set with balanced classes.** To quantify the effect of `class balance` in combination with `size`, we further created balanced sets containing 30k and 60k instances, respectively. **Furthermore, we perform hyper-parameter optimization on these data sets.** Finally, we decided to apply the `NN` classifier as well, since we want to know whether the finding of (Conneau et al., 2018a; Perone et al., 2018) (removing the hidden layer improves the performance on the WC task)

[56] (Conneau et al., 2018a) refer to the `NN` classifier as logistic regression. This implementation must not be confused with `LR` which we adopted from the *scikit-learn* library.

can be reproduced on balanced data. Figure 30 visualizes the absolute performance deltas in the various settings. We produced all results on the balanced 10k data set using the NN_H classifier. Results for 30k and 60k training examples can be found in appendix D on page 83 (cf. table 49 on page 85).

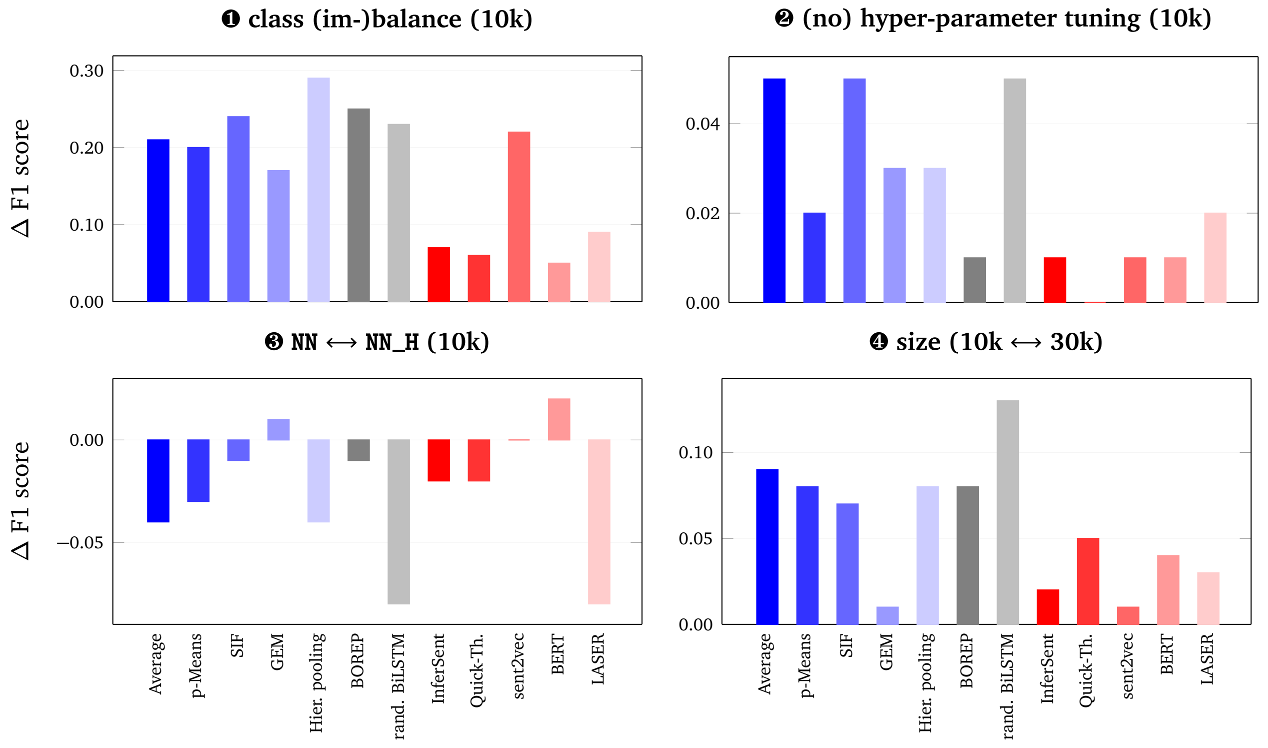


Figure 30: Absolute F1 performance deltas in the WC task (NN_H): ❶ Effect of `class balance` (positive number indicates positive effect of balanced data), ❷ Effect of `hyper-parameter tuning` (on balanced data), ❸ Effect of `classifier` (on balanced data; negative values indicate worse performance of NN) and ❹ effect of `size` (on balanced data sets).

❶ Effect of class (im-)balance. Figure 30 (top left image) shows that **a balanced data set has positive effects on the performance of all embeddings**. We realized immediately that compositional models like the *vanilla average* (Δ +0.21 F1), *hierarchical pooling* (Δ +0.29 F1) or *SIF* (Δ +0.24 F1) as well as random encoders profit the most from a balanced data set. Furthermore, the performance of *sent2vec* is boosted considerably as well (Δ +0.22 F1). The scores of more sophisticated embeddings like *InferSent* (Δ +0.07 F1), *LASER* (Δ +0.09 F1) or *Quick-Thought* (Δ +0.06 F1) are less influenced by a uniform class distribution and hardly change as a consequence. **The results suggest that the performance of averaging methods relies more on a balanced class distribution**. The performance improvement of *sent2vec* is plausible in this context, since it is closely related to compositional models. **Conversely, this means that trained encoder architectures are more robust with respect to changes in the class distribution.**[57] This knowledge is essential, given the fact that in real-world applications, classes are rarely perfectly balanced. Such circumstances require the usage of robust embeddings which are also capable of handling data belonging to rare classes gracefully. As table 27 on the following page shows, also the embedding ranking is influenced by the `class balance` factor. We computed a Spearman correlation of 0.91. Detailed results can be found in table 51 on page 86 in the appendix.

❷ Effect of hyper-parameter optimization. Analogously to (Conneau et al., 2018a), we implemented a hyper-parameter search over the parameters `# hidden units` $\in \{50, 100, 200\}$ and `dropout` $\in \{0, 0.1, 0.2\}$. The 10k, 30k and 60k data sets were split into respective `train`, `dev` and `test` portions, where we used the `dev` set for the hyper-parameter optimization. Additionally, in order to prevent overfitting, an **early-stopping mechanism** on the `dev` set was introduced with the `patience` parameter set to a value of 5 (Conneau and colleagues also use early stopping). Figure 30 (top right image) reveals that **hyper-parameter tuning does not have a considerable impact on the performance of the embeddings**: The largest improvement can be observed for compositional and random models (especially the *vanilla average*, *SIF* and *random BiLSTM* embeddings with Δ +0.05 F1 each). From table 27 on the next page we notice that the ranking of compositional models changes slightly, whereas the ranking of trained encoders is more robust. The Spearman correlation of the two rankings amounts to 0.95. Detailed results can be found in table 50 on page 85 in the appendix.

[57] At least in the WC task in the English language.

	Vanilla average	p-Means	SIF	GEM	hier. pooling	BOREP	Random BiLSTM	InferSent	Quick-Thought	sent2vec	BERT	LASER
❶ class (im-)balance												
Ranking *(imbalanced, 10k)*	8	6	7	5	12	9	11	1	3	4	9	2
Ranking *(balanced, 10k)*	8	7	6	5	11	9	10	2	4	1	12	3
	Spearman correlation: 0.91											
❷ (no) hyper-parameter tuning												
Ranking *(balanced, no optimization, 10k)*	8	7	6	5	11	9	10	2	4	1	12	3
Ranking *(balanced, optimization, 10k)*	7	8	4	4	11	10	9	2	6	1	12	3
	Spearman correlation: 0.95											
❹ size (10k ⟷ 30k)												
Ranking *(balanced, 10k)*	8	7	6	5	11	9	10	2	4	1	12	3
Ranking *(balanced, 30k)*	6	6	5	8	11	10	9	2	4	1	12	3
	Spearman correlation: 0.94											
❹ size (30k ⟷ 60k)												
Ranking *(balanced, 30k)*	6	6	5	8	11	10	9	2	4	1	12	3
Ranking *(balanced, 60k)*	7	5	5	9	11	10	8	1	4	1	12	3
	Spearman correlation: 0.98											

Table 27: Effect of various factors on the embedding ranking.

❸ Effect of using the `NN` classifier on balanced data sets. (Conneau et al., 2018a) as well as (Perone et al., 2018) stated that they removed the hidden layer for the WC task, since this produced better results. **This effect could not be replicated before and also on balanced data sets we cannot observe this effect**: As figure 30 on the previous page (bottom left image) shows, the `NN` classifier performs much worse for almost all embeddings (9 out of 12). However, this gap vanishes as the number of instances increases (cf. table 49 on page 85 in the appendix): For 60k training examples, the `NN` model is mostly on par with the `NN_H` classifier. Since Conneau and Perone used even more data, it is conceivable that the `NN` model might outperform the `NN_H` classier on even larger data sets. Furthermore, their task was set up using 1k classes which can have an influence as well. Otherwise, this effect may be specific to the data set used by Conneau and Perone. Detailed results can be found in table 51 on page 86 in the appendix.

❹ Effect of data set size on balanced data sets. Our results suggest that **the data set size does have a positive effect on the performance** (cf. figure 30 on the preceding page; bottom right image), yet it is not as strong as the effect of balancing the class distribution. Again, compositional methods and random encoders improve a bit more compared to trained encoder architectures. While the ranking changes slightly between 10k and 30k instances (correlation of 0.94), it is rather stable between 30k and 60k instances for which we computed a correlation of 0.98 (cf. table 27). This confirms the results obtained in the previous experiments, where a similar effect could be found. **This suggests that a probing task data set size of approximately 30k instances is sufficient to achieve a stable embedding ordering.** Detailed results can be found in table 51 on page 86 in the appendix.

The last column of table 50 on page 85 presents the final results obtained, if all additional effects are taken into account (using the `NN_H` classifier on a large and balanced data set with hyper-parameter tuning enabled). These results come much closer to the ones reported in the literature (**NB:** different amount of classes). As the table reveals, most embeddings exhibit a performance greater than or equal to 0.90 F1 score (except for random encoders, *BERT* and *hierarchical pooling*) which implies that **the worse performances in the previous sections are caused by a combination of multiple effects.**

Hyper-parameter optimization in other tasks. The following table 28 on the following page lists the results obtained by conducting a hyper-parameter optimization on three other tasks, namely SENTI, VOICE and SUBJNUM. This was done in order to exclude the possibility that the negligible effect of hyper-parameter tuning we found for WC is specific to this task. **However, the table confirms the low impact of optimizing the hyper-parameters also in other tasks:** *InferSent* profits the most on the SENTI task (Δ +0.05 F1 score); on VOICE it is *LASER* and *InferSent* (Δ +0.03 F1 score each), while *Quick-Though* can improve the most on the SUBJNUM task (Δ +0.04 F1 score).

Effect of hyper-parameter tuning on other tasks (original data sets)									
Task	**Senti**			**Voice**			**SubjNum**		
Tuning	**yes**	**no**	**Δ**	**yes**	**no**	**Δ**	**yes**	**no**	**Δ**
Vanilla average	0.72	0.70	**0.02**	0.80	0.80	**0.00**	0.86	0.86	**0.00**
p-Means	0.70	0.70	**0.00**	0.83	0.81	**0.02**	0.88	0.85	**0.03**
BOREP	0.65	0.65	**0.00**	0.80	0.78	**0.02**	0.87	0.84	**0.03**
Random BiLSTM	0.69	0.69	**0.00**	0.84	0.83	**0.01**	0.88	0.88	**0.00**
InferSent	0.75	0.70	**0.05**	0.90	0.87	**0.03**	0.91	0.88	**0.03**
Quick-Thought	0.67	0.64	**0.03**	0.83	0.81	**0.02**	0.80	0.76	**0.04**
sent2vec	0.66	0.66	**0.00**	0.78	0.78	**0.00**	0.77	0.76	**0.01**
BERT	0.65	0.64	**0.01**	0.78	0.76	**0.02**	0.88	0.86	**0.02**
LASER	0.74	0.73	**0.01**	0.93	0.90	**0.03**	0.92	0.90	**0.02**

Table 28: Effect of hyper-parameter tuning in other tasks (measured by F1 score).

Summary

Large (at least 30k examples) and balanced data sets have a substantial positive effect on the performance of the embeddings. Hyper-parameter tuning, on the other hand, was found to be of minor importance (observed on several tasks). Overall, we found that compositional models profit the most. The results furthermore suggest to use neural architectures for the evaluation, since such models consistently outperform other classifiers in terms of absolute scores.

7.6 Discussion, Limitations and Recommendations for future Work

Research question ❶. The results of the analyses presented in this chapter demonstrate that **trained encoders are superior to compositional models or random encoders in English**. Nevertheless, as (Wieting and Kiela, 2019) already found out, random encoders provide a very strong baseline and the gap to trained architectures is often smaller than desirable. **This gap vanishes even more in low-resource languages:** A lack of sufficient amounts of training data on which the encoder architectures can be trained, ultimately entails worse performance. As a consequence, compositional embeddings and random encoders are sometimes able to outperform trained models in such settings. Despite worse absolute scores, the top-three analysis reveals that trained embeddings can still have advantages if resources are scarce. Also, in order to check whether low-quality translations have a considerable influence on the performance in low-resource languages, it might be worthwhile to check their trustworthiness. The translated sentences could be translated back into English to see whether or not the results differ substantially from the original sentences.

The results obtained in this work further confirm that, independently of the language, still no universal sentence embedding exists: The performance of the embeddings strongly depends on the task at hand which means that there is no embedding which consistently shows strong performance (Perone et al., 2018). *LASER*, however, scores highly across many tasks and languages and can therefore be considered close to a universal sentence embedding. It has the advantage that it was trained on high-quality and multilingual corpora which did not have to be translated. Also *InferSent* shows convincing performances for the English language. In the other languages, however, its scores should be treated with caution, since the models were not trained on the full *AllNLI* data set which impairs the performance. Therefore, we should continue the translation process and retrain the models, once the full data set is available in all target languages.

Surprisingly, we found that high-dimensional embeddings do not necessarily have advantages in the evaluation on probing tasks. Negative correlations between embedding dimensionality and performance in some of the probing tasks suggest that low-dimensional embeddings can achieve better scores than high-dimensional ones. **Also, we notice that there is much more variance in the probing task performances in comparison to the downstream task results.** We hold the artificial setup of probing tasks accountable for this circumstance. This result can also be found in the paper by (Perone et al., 2018).

The correlations computed between probing and downstream task results (cf. section 7.4 on page 56) suggest for the English language that the **knowledge about word containment is a strong predictor of downstream task performance**. Also VOICE is correlated positively with most of the downstream applications considered. In this context we found some inconsistencies with results in the literature. However, (Eger et al., 2019) report that the correlations depend on the set of embeddings considered in the analysis. Additionally, we detected that the language has an influence on the ultimate results: **The correlations for the other target languages behave differently compared to English:** The TREC task for which we computed high positive correlations in English does not have substantial connections to the probing tasks in the remaining languages. Also, we found many negative values. The strongest negative correlation could be found for the Georgian WO task. We believe that the flexible word order in the Georgian language is responsible for this.

Research question ❷. Different setups in the literature and our work caused discrepancies which we attempted to analyze more in depth: **One of the main findings in this context was that it is crucial to evaluate the embeddings on balanced probing task data sets** (cf. section 7.5 on page 60). Especially compositional models were found to profit from such a setting and as a consequence improve up to almost 0.30 F1 score (cf. *hierarchical pooling*). Imbalanced data sets therefore cause a distortion between compositional and trained sentence embedding models (the rankings change considerably). Naturally, the choice of using balanced or imbalanced data sets depends on the research questions to be addressed. In reality, data sets are rarely balanced, i. e. if the embeddings are supposed to be tested in a more realistic scenario, the data sets would have to be imbalanced. However, probing tasks were especially designed to test sentence representations in a laboratory environment which assumes perfect conditions. In addition, the size of the probing task data set matters: **We detected that at least 30k instances should be used in order to get a stable embedding ordering.** **The stability analysis suggests that our results should be treated with caution:** The results reported in this work were produced on 10k data sets which are partially imbalanced. The reason for this is that it turned out to be very difficult to create large data sets for the low-resource languages. In the interest of comparability, we decided to limit the size of the data sets to 10k also for the English language. In future work, more effort should be put into the creation of larger and balanced probing task data sets.

While the factors `class balance` and `size` have substantial influence on the performance on probing tasks, we found that **`hyper-parameter` tuning only accounts for minor improvements** which could be replicated in several tasks. Finally, the ordering of the embeddings is partially very unstable across different classifiers but neural architectures consistently outperform other models like random forests in terms of absolute scores. **Finally, we give the following recommendations for a good probing task setup in future work:**

Recommendations for future work

1. Probing task data sets should comprise at least **30k instances**.
2. The data sets should be **balanced**.
3. The probing task classifier should be a **neural network** (preferably an MLP).
4. Optionally, hyper-parameter tuning can be used despite of only slight positive effects (it does not hurt the performance).

Some of the recommendations, especially the creation of large and balanced data sets, are challenging to implement for low-resource languages like Georgian. Fortunately, additional *UD* tree banks for the Georgian language will be available soon. This data can be used to create even better data sets for the probing tasks. Despite the detailed analysis, some of the discrepancies still remain: E. g. the performance of the *power means* embedding on the WC task is still lower compared to (Perone et al., 2018), despite less classes in our setup. Explanations for this circumstance are speculative and require further analysis (even larger data sets?, embedding dimension?). Also, we still notice differences on the TREC task. **In order to enable a smoother comparison with the literature, we recommend that future work in this domain makes use of the *SentEval* framework.** Furthermore, we have to make sure that the embeddings to be tested are produced using the same hyper-parameter settings as in the literature (same embedding dimensionalities, window sizes, ...).

The evaluation setups in the literature mostly respect the recommendations given above and the setup we implemented should have been more oriented towards the evaluation protocol used in the literature. Nonetheless, the stability analysis clearly demonstrates that the the ultimate evaluation results depend on the setting to a large extent. **We posit that the evaluation should be agnostic with respect to external factors like class balance or the number of training examples. Since the encoder architectures are frozen, the representations for single sentences do not change as a consequence of larger or balanced probing task data sets. Therefore, it is not entirely clear to us, what the probing task results indicate. We hope that future work in this area will provide clarification.**

7.7 Summary

This chapter provided details about the evaluation results on probing and downstream tasks. Section 7.2 on page 46 first of all presented the probing task results while the subsequent section 7.3 on page 53 focused on downstream task performances. As soon as the embeddings were evaluated intrinsically (probing tasks) as well as extrinsically (downstream tasks), we could correlate the results in order to get clarity about which linguistic properties are decisive for downstream tasks performance (cf. section 7.4 on page 56). It was shown that the correlations vary between the languages.

Since the evaluation is not performed consistently in the literature and also to investigate discrepancies in the context of this work, an additional stability analysis was performed (cf. section 7.5 on page 60). In this experiment, various factors (data set `size`, `class balance`, `classifier` architectures and `hyper-parameter optimization`) were changed systematically and the influence on the performances was recorded. Based on the outcomes of this analysis, we gave recommendations w. r. t. a better evaluation setup for future work (cf. section 7.6 on page 66)

8 Conclusion

The goal of the present thesis was focused towards the interpretability of sentence representations in low-resource languages. **A review of the current literature concerning the evaluation of sentence embeddings showed that the analyses in this domain are mainly done for English and other high-resource languages, very often omitting low-resource languages entirely. This is why the main goal of this work was to extend the probing task setting to lower-resource languages in order to investigate whether findings reported for the English language are reproducible for languages with less resources.** Next to English and German, we chose to include the low-resource languages Russian, Turkish and Georgian. The following paragraphs summarize the main aspects of the thesis:

Structure of the thesis. Section 1 on page 1 first of all gave a short introduction into the topic of this thesis and introduced the research questions which are addressed in the context of this work. The subsequent section 2 on page 3 demonstrated which efforts have already been made with respect to the development of word and sentence embedding algorithms and gave furthermore information about current evaluation methodologies which are leveraged for the interpretation of what aspects of sentences are retained by sentence embeddings. Sections 3 on page 7 and 4 on page 12 then introduced some of the word and sentence embedding algorithms in depth which were used in the subsequent evaluations. As a next step, section 5 on page 21 presented the notion of probing tasks as well as a list of tasks already introduced in the literature. **These tasks are generally divided into three categories: tasks probing for superficial, syntactical or semantic aspects of sentences**. In order to facilitate the selection of specific probing tasks for the low-resource target languages, a short presentation of Russian, Turkish and Georgian was given with respect to their grammatical idiosyncrasies. This was necessary, since some linguistic properties may not be present in some of the languages, rendering probing for such properties useless (e. g. recall that there are no articles in the Turkish language). After we had made a choice for a set of probing tasks, we provided more information about the process of data set creation and presented which corpora we used as a data source. *UD* offers tree banks for many languages covering diversified styles of writing, which is why we chose them for the task at hand. For Georgian it was necessary to resort to the *GNC* corpus, since no *UD* corpus exists (but soon will). Analogously, chapter 6 on page 41 presented the three downstream applications: Argumentation mining, sentiment analysis and question type detection. Finally, chapter 7 on page 46 introduced the evaluation results obtained on the probing and downstream tasks. Additionally, the results of the stability analysis were discussed and recommendations for future work were given. The main results are as follows:

Main results. The results indicate that **trained architectures have advantages over compositional models in high-resource languages in terms of absolute performance**. In low-resource languages, however, compositional models catch up and sometimes manage to outperform their trained counterparts. Nevertheless, the top-three analysis reveals that trained architectures can still be advantageous in low-resource languages. Random encoders furthermore surprise with unexpected high evaluation scores. The correlations computed between probing and downstream task performances were found to be unstable across different languages. Compared to English, we found more correlations falling below an absolute value of 0.20 in German, Russian, Turkish and Georgian. Many correlations even fell into the negative range, where especially the Georgian WO task is concerned (due to flexible word order). Since a comparison of our results with the literature revealed inconsistencies in the evaluation setups, we performed an additional stability analysis to investigate the origins of these divergences. Several scenarios (e. g. balanced ⟷ imbalanced data sets, hyper-parameter tuning ⟷ no hyper-parameter tuning, etc.) were tested and the performance of the embeddings was recorded. This way it was possible to determine which factors have the strongest influence on the performance of the embeddings and hence on their relative ordering. **The overall results of this analysis suggest that the findings reported in earlier sections have to be treated with caution**, since we found that data sets with less than 30k training instances induce a rather unstable embedding ordering, while the rankings do not change substantially with more than 30k instances. Furthermore, balanced data sets led to large improvements, especially among the compositional models. Hyper-parameter tuning, on the other hand, did not yield substantially different results and improved the performances only slightly.

Future work. Future research on the evaluation of sentence embeddings in low-resource languages should follow the recommendations we gave in the previous section. The usage of larger (at least 30k instances) and balanced data sets is necessary to get more reliable results. Furthermore, we recommend to use the *SentEval* framework for future experiments. The stability analysis made above all clear that the results vary substantially in different setups. **We posit that factors like data set size and class balance should have no influence on the evaluation results. The results lead us to believe that the evaluation on probing tasks is not the optimal strategy. Hopefully, additional research can bring more clarity on that. Also we hope that future work will put lower-resource languages more in focus. As some of the results show, patterns for the English language are not necessarily generalizable to other languages.** Finally, we release our implementation for future experiments: `https://git.ukp.informatik.tu-darmstadt.de/UKP-Students/interpretability_sentence_embeddings`

A Probing Task Setup

EN	DE	RU	TR	KA
project	Insel	миллионов	vergi	გაეროს
horse	Norden	информации	yüzde	პერიოდში
dinner	Bevölkerung	территории	karşılık	მოსახლეობის
countries	Partei	рынке	Devlet	ხალხის
restaurant	Arten	программы	sırasında	თავმჯდომარე
back	Stunden	уровень	makine	შეხვედრა
women	Zahl	начала	içeriye	ამერიკის
Percentage	Mannschaft	производства	okula	კომისიის
moment	Tochter	степени	mektup	წელი
changes	Provinz	период	haber	წევრი
minutes	Laden	ходе	sahibi	პარლამენტში
members	Betrieb	средства	Kuzey	ოპოზიციის
security	Beginn	основе	cevap	განცხადება
reason	Bedeutung	граждан	şirketi	რაიონის
email	Süden	отношения	yüzyılda	განვითარების
language	Wochen	движения	yüzünden	პოლიციის
peace	Westen	данным	dönemde	მიზეზი
position	Ergebnis	большинство	durumunda	პროცესი
girl	Minuten	безопасности	yola	წლებში
PivotTable	Dorf	участие	öğrenci	გუნდი
level	Länge	взгляд	çocuğun	თამაში
city	Vertrag	культуры	sorun	კავშირის
agreement	Jahrhundert	группы	oğlum	მიზნით
table	Ortsteil	директор	seçim	წლების
front	Landkreis	проект	isim	გუნდის
control	Hilfe	армии	doğum	საშუალება
others	Verein	положение	ilişki	მინისტრმა
thanks	Hälfte	действия	masanın	წარმომადგენლები
attention	Verbindung	целом	gözleri	სახით
hotel	Zeitpunkt	смысле	ayında	ქალაქის

Table 29: List of the 30 chosen mid-frequency nouns for each language (WC task). The table entries are sorted by frequencies.

EN	DE	RU	TR	KA
know	sein	знать	yapmak	ყოფნა
speak	tun	жить	gitmek	ქონა
think	gehen	любить	gelmek	ყოლა
understand	machen	работать	almak	სვლა
remember	wissen	ждать	istemek	ვხედავ
eat	leben	говорить	çalışmak	ყურება
happen	arbeiten	думать	bilmek	მოსმენა (listen)
get	warten	понимать	konuşmak	მოსმენა (hear)
come	sprechen	мочь	okumak	გაგება
avoid	denken	хотеть	sevmek	ცოდნა
pretend	verstehen	делать	demek	ფიქრი
mitigate	wollen	брать	düşünmek	კეთება
pray	nehmen	давать	yemek	ლაპარაკი
forgive	geben	помнить	içmek	თქმა
	sagen		başlamak	საუბარი
	fragen		olmak	მოყოლა
	trinken		söylemek	წდომა
	essen		yatmak	ცხოვრება
	passieren		oturmak	სიკვდილი
	mögen		sormak	ჭამა
	fliegen		inanmak	სმა
	haben		açıklamak	ცეკვა
			açmak	ძილი
			affetmek	მიცემა
			almak	აღება
			anlamak	
			aramak	
			ağlamak	
			abartmak	
			beğenmek	
			beklemek	
			atlamak	
			atmak	
			azaltmak	
			korkmak	
			koşmak	
			kullanmak	
			önermek	

Table 30: List of the chosen verbs for each language (SVAgree task).

Remarks concerning the tables in appendix B on the next page and C on page 79:
The following tables list the results on probing tasks and downstream tasks in detail. Next to the test scores, the tables furthermore contain the relative rankings of the embeddings per task. These rankings can be found in the columns labeled with 'R'. In order to facilitate a quick overview, the top-three performances were highlighted: A golden cell background indicates rank 1, silver color rank 2, and finally, a bronze color denotes rank 3. The results were produced without hyper-parameter optimization.

B Detailed Probing Task Results

B.1 Results measured by F1 Score

Language: English (F1)																		
Embedding	Surface Tasks				Syntactic Tasks										Semantic Tasks			
	SentLen		WC		BiShift		SVAgree		SVDist		Voice		WO		EOS		SubjNum	
	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R
Size of the data set																		
# instances	10,000	-	10,000	-	10,000	-	10,000	-	10,000	-	9,999	-	10,000	-	10,000	-	9,448	-
Majority class (baseline)																		
Majority	17.9	-	13.3	-	50.2	-	51.1	-	47.5	-	61.0	-	33.4	-	21.7	-	75.7	-
Random embeddings (baseline)																		
BOREP	00.438	10	00.551	18	00.488	18	00.737	18	00.279	10	00.776	10	00.188	15	00.213	10	00.844	7
Random BiLSTM	00.549	7	00.462	20	00.510	6	00.819	12	00.323	5	00.833	3	00.598	4	00.246	3	00.879	2
Average embeddings (*FastText*)																		
Vanilla average	00.240	17	00.581	17	00.485	20	00.775	16	00.265	13	00.798	7	00.201	11	00.150	19	00.855	5
p-Means	00.553	5	00.657	13	00.491	15	00.804	14	00.309	6	00.807	6	00.173	19	00.231	6	00.850	6
SIF	00.217	19	00.634	14	00.471	21	00.808	13	00.256	15	00.756	15	00.208	10	00.144	20	00.835	8
GEM	00.301	13	00.778	10	00.496	11	00.776	15	00.300	8	00.718	19	00.068	22	00.206	12	00.797	10
hier. pooling	00.454	9	00.354	22	00.501	9	00.660	21	00.271	12	00.779	8	00.363	8	00.232	5	00.817	9
Average embeddings (*word2vec*)																		
Vanilla average	00.243	16	00.803	7	00.497	10	00.892	6	00.255	16	00.760	13	00.187	16	00.172	16	00.659	17
p-Means	00.503	8	00.822	5	00.502	8	00.939	1	00.295	9	00.764	12	00.183	18	00.236	4	00.656	18
SIF	00.215	20	00.809	6	00.506	7	00.889	8	00.248	18	00.741	17	00.191	14	00.158	18	00.619	21
GEM	00.256	14	00.597	16	00.494	13	00.710	20	00.241	22	00.684	21	00.111	21	00.179	15	00.573	22
hier. pooling	00.355	12	00.695	12	00.493	14	00.841	10	00.253	17	00.724	18	00.233	9	00.224	8	00.633	20
Average embeddings (*Attract-Repel*)																		
Vanilla average	00.183	21	00.622	15	00.491	15	00.890	7	00.247	19	00.753	16	00.198	13	00.141	21	00.738	14
p-Means	00.585	4	00.946	1	00.464	22	00.936	2	00.279	10	00.818	4	00.186	17	00.210	11	00.761	11
SIF	00.162	22	00.737	11	00.488	18	00.903	5	00.243	21	00.699	20	00.200	12	00.130	22	00.685	16
GEM	00.230	18	00.803	7	00.491	15	00.743	17	00.259	14	00.650	22	00.148	20	00.167	17	00.640	19
hier. pooling	00.419	11	00.406	21	00.496	11	00.835	11	00.245	20	00.770	11	00.370	7	00.193	13	00.700	15
Trained embeddings																		
InferSent	00.600	3	00.919	2	00.536	4	00.912	3	00.414	2	00.868	2	00.760	3	00.230	7	00.879	2
Quick-Thought	00.649	2	00.848	3	00.544	3	00.887	9	00.410	3	00.811	5	00.802	2	00.221	9	00.758	13
sent2vec	00.254	15	00.786	9	00.534	5	00.910	4	00.301	7	00.779	8	00.381	6	00.191	14	00.759	12
BERT	00.551	6	00.531	19	00.641	1	00.718	19	00.406	4	00.760	13	00.573	5	00.282	2	00.858	4
LASER	00.697	1	00.832	4	00.600	2	00.572	22	00.521	1	00.904	1	00.846	1	00.319	1	00.899	1

Table 31: Probing task results for the English language (F1 scores).

Language: German (F1)																		
Embedding	Surface Tasks				Syntactic Tasks										Semantic Tasks			
	SentLen		WC		BiShift		SVAgree		SVDist		Voice		WO		EOS		SubjNum	
	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R
Size of the data set																		
# instances	10,000	-	10,000	-	10,000	-	10,000	-	10,000	-	9,999	-	10,000	-	10,000	-	9,999	-
Majority class (baseline)																		
Majority	19.9	-	14.8	-	50.2	-	50.7	-	43.6	-	50.0	-	33.4	-	19.3	-	50.0	-
Random embeddings (baseline)																		
BOREP	00.402	10	00.668	7	00.482	22	00.846	2	00.327	10	00.790	17	00.173	22	00.205	11	00.858	15
Random BiLSTM	00.508	4	00.577	8	00.490	16	00.847	1	00.399	4	00.864	4	00.735	2	00.229	8	00.895	7
Average embeddings (*FastText*)																		
Vanilla average	00.195	15	00.695	6	00.489	18	00.822	10	00.318	14	00.832	12	00.215	16	00.172	15	00.903	3
p-Means	00.492	6	00.745	4	00.489	18	00.837	5	00.336	9	00.856	7	00.199	20	00.232	6	00.890	9
SIF	00.164	18	00.737	5	00.500	10	00.813	12	00.293	17	00.793	15	00.216	14	00.153	19	00.898	5
GEM	00.249	14	00.816	3	00.512	5	00.830	7	00.325	11	00.854	8	00.222	12	00.198	12	00.889	10
hier. pooling	00.433	9	00.524	10	00.492	13	00.794	15	00.322	13	00.762	20	00.451	6	00.217	9	00.833	16
Average embeddings (*word2vec*)																		
Vanilla average	00.170	17	00.296	13	00.491	15	00.826	8	00.337	8	00.860	5	00.226	11	00.154	18	00.891	8
p-Means	00.482	7	00.255	15	00.500	10	00.818	11	00.325	11	00.858	6	00.174	21	00.231	7	00.883	11
SIF	00.143	21	00.284	14	00.504	7	00.800	14	00.312	15	00.839	10	00.220	13	00.141	21	00.880	13
GEM	00.185	16	00.157	19	00.503	8	00.640	21	00.254	20	00.664	22	00.205	19	00.162	16	00.650	22
hier. pooling	00.280	12	00.242	16	00.492	13	00.790	17	00.311	16	00.819	13	00.249	10	00.192	14	00.865	14
Average embeddings (*Attract-Repel*)																		
Vanilla average	00.156	20	00.154	20	00.495	12	00.808	13	00.261	19	00.798	14	00.216	14	00.149	20	00.808	18
p-Means	00.500	5	00.238	17	00.487	20	00.840	4	00.339	7	00.877	1	00.211	17	00.237	5	00.828	17
SIF	00.128	22	00.130	21	00.486	21	00.793	16	00.224	22	00.764	19	00.209	18	00.123	22	00.791	19
GEM	00.162	19	00.185	18	00.502	9	00.663	20	00.239	21	00.744	21	00.250	9	00.161	17	00.741	21
hier. pooling	00.342	11	00.128	22	00.490	16	00.774	18	00.270	18	00.781	18	00.324	8	00.211	10	00.759	20
Trained embeddings																		
InferSent	00.557	2	00.449	11	00.505	6	00.834	6	00.367	6	00.838	11	00.713	3	00.242	4	00.882	12
Quick-Thought	00.520	3	00.346	12	00.575	3	00.825	9	00.456	1	00.868	3	00.490	5	00.269	2	00.901	4
sent2vec	00.271	13	00.968	1	00.521	4	00.845	3	00.370	5	00.874	2	00.429	7	00.197	13	00.906	1
BERT	00.467	8	00.577	8	00.636	1	00.771	19	00.404	3	00.793	15	00.584	4	00.269	2	00.896	6
LASER	00.608	1	00.861	2	00.606	2	00.639	22	00.448	2	00.854	8	00.821	1	00.307	1	00.905	2

Table 32: Probing task results for the German language (F1 scores).

Language: Russian (F1)																		
Embedding	Surface Tasks				Syntactic Tasks										Semantic Tasks			
	SentLen		WC		BiShift		SVAgree		SVDist		Voice		WO		EOS		SubjNum	
	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R
Size of the data set																		
# instances	10,000	-	10,000	-	10,000	-	10,000	-	10,000	-	9,999	-	10,000	-	10,000	-	9,999	-
Majority class (baseline)																		
Majority	17.4	-	09.7	-	51.3	-	50.1	-	42.3	-	68.0	-	33.4	-	23.6	-	50.0	-
Random embeddings (baseline)																		
BOREP	00.405	10	00.515	14	00.492	15	00.752	13	00.231	10	00.882	6	00.177	19	00.209	10	00.863	9
Random BiLSTM	00.515	6	00.552	13	00.514	5	00.773	5	00.272	4	00.918	1	00.716	3	00.231	6	00.901	3
Average embeddings (*FastText*)																		
Vanilla average	00.208	15	00.672	12	00.483	18	00.754	10	00.216	14	00.883	5	00.213	14	00.125	20	00.913	2
p-Means	00.519	5	00.673	11	00.474	20	00.771	6	00.256	7	00.912	2	00.202	17	00.239	4	00.899	4
SIF	00.193	18	00.694	10	00.494	14	00.754	10	00.213	17	00.848	11	00.212	15	00.126	19	00.916	1
GEM	00.257	14	00.743	9	00.501	8	00.745	14	00.226	11	00.895	3	00.167	20	00.197	13	00.876	7
hier. pooling	00.456	8	00.443	16	00.499	9	00.707	15	00.223	12	00.856	10	00.406	6	00.215	9	00.851	11
Average embeddings (*word2vec*)																		
Vanilla average	00.207	16	00.947	3	00.498	10	00.795	3	00.222	13	00.844	13	00.208	16	00.168	14	00.851	11
p-Means	00.509	7	00.990	1	00.497	11	00.800	2	00.240	8	00.862	9	00.166	21	00.226	7	00.838	15
SIF	00.163	19	00.937	4	00.497	11	00.779	4	00.214	15	00.819	15	00.220	12	00.147	18	00.851	11
GEM	00.154	20	00.803	7	00.504	7	00.653	17	00.214	15	00.619	17	00.162	22	00.151	17	00.633	19
hier. pooling	00.359	11	00.854	6	00.497	11	00.761	9	00.235	9	00.827	14	00.241	9	00.208	11	00.810	16
Average embeddings (*Attract-Repel*)																		
Vanilla average	00.075	22	00.295	20	00.456	22	00.505	22	00.195	20	00.502	21	00.225	10	00.083	22	00.635	17
p-Means	00.452	9	00.398	17	00.481	19	00.507	21	00.197	19	00.619	17	00.185	18	00.234	5	00.631	20
SIF	00.091	21	00.294	21	00.469	21	00.514	19	00.193	22	00.490	22	00.219	13	00.089	21	00.634	18
GEM	00.328	12	00.294	21	00.488	16	00.510	20	00.207	18	00.585	19	00.223	11	00.204	12	00.615	22
hier. pooling	00.201	17	00.320	19	00.488	16	00.516	18	00.194	21	00.557	20	00.247	8	00.157	16	00.628	21
Trained embeddings																		
InferSent	00.576	2	00.389	18	00.530	4	00.753	12	00.264	6	00.891	4	00.651	4	00.225	8	00.891	5
Quick-Thought	00.574	3	00.904	5	00.550	3	00.768	8	00.344	3	00.846	12	00.727	2	00.253	2	00.842	14
sent2vec	00.273	13	00.968	2	00.506	6	00.822	1	00.269	5	00.871	8	00.351	7	00.165	15	00.876	7
BERT	00.540	4	00.495	15	00.597	1	00.673	16	00.375	2	00.802	16	00.585	5	00.252	3	00.852	10
LASER	00.636	1	00.776	8	00.574	2	00.769	7	00.496	1	00.877	7	00.779	1	00.294	1	00.878	6

Table 33: Probing task results for the Russian language (F1 scores).

Language: Turkish (F1)																		
Embedding	Surface Tasks				Syntactic Tasks										Semantic Tasks			
	SentLen		WC		BiShift		SVAgree		SVDist		Voice		WO		EOS		SubjNum	
	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R
Size of the data set																		
# instances	10,000	-	10,000	-	10,000	-	10,000	-	2,750	-	8,416	-	10,000	-	10,000	-	4,030	-
Majority class (baseline)																		
Majority	37.3	-	12.6	-	50.5	-	50.6	-	39.0	-	86.2	-	33.4	-	23.7	-	83.4	-
Random embeddings (baseline)																		
BOREP	00.407	10	00.673	18	00.486	22	00.599	20	00.411	11	00.661	12	00.162	19	00.328	12	00.703	9
Random BiLSTM	00.514	5	00.591	19	00.553	5	00.712	4	00.437	6	00.703	3	00.656	3	00.382	4	00.736	7
Average embeddings (*FastText*)																		
Vanilla average	00.174	18	00.705	16	00.487	20	00.631	17	00.359	15	00.682	6	00.203	13	00.264	18	00.752	3
p-Means	00.511	6	00.761	13	00.487	20	00.642	15	00.439	5	00.690	4	00.161	20	00.353	9	00.744	5
SIF	00.158	20	00.738	14	00.492	17	00.643	14	00.352	16	00.689	5	00.203	13	00.249	20	00.752	3
GEM	00.276	13	00.798	10	00.503	9	00.613	19	00.380	13	00.710	2	00.090	22	00.305	13	00.759	2
hier. pooling	00.419	9	00.516	20	00.507	6	00.557	22	00.420	9	00.627	18	00.322	7	00.356	7	00.697	10
Average embeddings (*word2vec*)																		
Vanilla average	00.192	17	00.873	4	00.496	13	00.699	7	00.345	17	00.659	14	00.190	15	00.279	16	00.621	12
p-Means	00.471	8	00.916	2	00.504	8	00.695	11	00.425	8	00.667	10	00.166	18	00.342	10	00.617	14
SIF	00.170	19	00.873	4	00.492	17	00.699	7	00.338	18	00.656	15	00.211	10	00.265	17	00.617	14
GEM	00.202	15	00.685	17	00.495	14	00.593	21	00.323	19	00.576	22	00.105	21	00.250	19	00.587	20
hier. pooling	00.358	12	00.840	7	00.498	12	00.640	16	00.412	10	00.639	16	00.290	8	00.336	11	00.602	19
Average embeddings (*Attract-Repel*)																		
Vanilla average	00.139	21	00.768	12	00.500	11	00.699	7	00.298	22	00.625	19	00.212	9	00.245	22	00.605	18
p-Means	00.527	3	00.936	1	00.493	15	00.699	7	00.442	4	00.681	7	00.172	17	00.369	5	00.608	17
SIF	00.127	22	00.833	8	00.506	7	00.710	5	00.300	21	00.622	20	00.205	12	00.247	21	00.612	16
GEM	00.196	16	00.783	11	00.490	19	00.629	18	00.302	20	00.590	21	00.188	16	00.281	15	00.580	22
hier. pooling	00.395	11	00.722	15	00.501	10	00.645	13	00.399	12	00.634	17	00.360	6	00.358	6	00.584	21
Trained embeddings																		
InferSent	00.522	4	00.447	22	00.565	4	00.658	12	00.432	7	00.670	9	00.650	4	00.398	2	00.708	8
Quick-Thought	00.504	7	00.867	6	00.587	2	00.737	2	00.448	3	00.660	13	00.768	1	00.356	7	00.621	12
sent2vec	00.237	14	00.891	3	00.493	15	00.701	6	00.367	14	00.663	11	00.210	11	00.288	14	00.663	11
BERT	00.583	2	00.494	21	00.585	3	00.725	3	00.454	2	00.676	8	00.585	5	00.396	3	00.744	5
LASER	00.615	1	00.805	9	00.596	1	00.774	1	00.536	1	00.768	1	00.743	2	00.472	1	00.803	1

Table 34: Probing task results for the Turkish language (F1 scores).

Language: Georgian (F1)														
Embedding	Surface Tasks				Syntactic Tasks								Semantic Tasks	
	SentLen		WC		BiShift		SVAgree		Voice		WO		EOS	
	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R
Size of the data set														
# instances	9,989	-	10,000	-	9,993	-	10,000	-	10,000	-	10,000	-	10,000	-
Majority class (baseline)														
Majority	28.3	-	13.4	-	51.5	-	50.8	-	65.1	-	33.5	-	28.2	-
Random embeddings (baseline)														
BOREP	00.373	11	00.519	15	00.491	10	00.655	2	00.769	4	00.125	21	00.348	9
Random BiLSTM	00.487	5	00.384	19	00.503	4	00.663	1	00.788	1	00.525	5	00.368	5
Average embeddings (*FastText*)														
Vanilla average	00.140	18	00.466	16	00.484	17	00.634	6	00.724	13	00.195	17	00.241	18
p-Means	00.498	3	00.564	13	00.487	13	00.639	4	00.782	2	00.139	20	00.366	6
SIF	00.120	20	00.461	17	00.485	16	00.626	8	00.698	16	00.198	16	00.222	20
GEM	00.291	13	00.704	4	00.491	10	00.630	7	00.755	8	00.123	22	00.316	13
hier. pooling	00.397	9	00.339	20	00.497	6	00.624	9	00.768	5	00.276	8	00.344	10
Average embeddings (*word2vec*)														
Vanilla average	00.177	17	00.578	10	00.492	9	00.563	20	00.692	17	00.252	9	00.249	16
p-Means	00.446	8	00.861	1	00.489	12	00.597	11	00.720	14	00.167	19	00.354	8
SIF	00.124	19	00.577	11	00.496	7	00.580	17	00.672	19	00.243	10	00.230	19
GEM	00.185	16	00.672	8	00.486	14	00.575	19	00.656	21	00.202	15	00.247	17
hier. pooling	00.359	12	00.701	5	00.484	17	00.581	16	00.731	11	00.363	7	00.317	12
Average embeddings (*Attract-Repel*)														
Vanilla average	00.107	21	00.541	14	00.463	22	00.594	12	00.677	18	00.225	11	00.213	21
p-Means	00.488	4	00.858	2	00.486	14	00.609	10	00.755	8	00.186	18	00.360	7
SIF	00.097	22	00.606	9	00.481	19	00.594	12	00.670	20	00.214	12	00.194	22
GEM	00.197	15	00.698	6	00.494	8	00.546	22	00.621	22	00.210	13	00.264	15
hier. pooling	00.390	10	00.566	12	00.475	20	00.591	14	00.744	10	00.366	6	00.319	11
Trained embeddings														
InferSent	00.537	2	00.262	22	00.471	21	00.635	5	00.771	3	00.543	4	00.393	3
Quick-Thought	00.454	7	00.679	7	00.528	3	00.579	18	00.707	15	00.787	1	00.387	4
sent2vec	00.240	14	00.799	3	00.501	5	00.650	3	00.726	12	00.206	14	00.294	14
BERT	00.472	6	00.446	18	00.586	1	00.549	21	00.761	7	00.584	3	00.413	2
LASER	00.705	1	00.270	21	00.533	2	00.590	15	00.765	6	00.640	2	00.435	1

Table 35: Probing task results for the Georgian language (F1 scores).

Top-three counts					
Embedding	**EN**	**DE**	**RU**	**TR**	**KA**
Vanilla Average	**0** (0.00)	**1** (0.11)	**3** (0.33)	**1** (0.11)	**0** (0.00)
p-Means	**3** (0.33)	**1** (0.11)	**3** (0.33)	**3** (0.33)	**4** (0.57)
SIF	**0** (0.00)	**0** (0.00)	**1** (0.11)	**1** (0.11)	**0** (0.00)
GEM	**0** (0.00)	**1** (0.11)	**1** (0.11)	**2** (0.22)	**0** (0.00)
hier. Pooling	**0** (0.00)	**0** (0.00)	**0** (0.00)	**0** (0.00)	**0** (0.00)
BOREP	**0** (0.00)	**1** (0.11)	**0** (0.00)	**0** (0.00)	**1** (0.14)
Random BiLSTM	**3** (0.33)	**2** (0.22)	**3** (0.33)	**2** (0.22)	**2** (0.29)
InferSent	**7** (0.78)	**2** (0.22)	**1** (0.11)	**1** (0.11)	**3** (0.43)
Quick-Thought	**5** (0.56)	**5** (0.56)	**5** (0.56)	**4** (0.44)	**2** (0.29)
sent2vec	**0** (0.00)	**4** (0.44)	**2** (0.22)	**1** (0.11)	**2** (0.29)
BERT	**2** (0.22)	**3** (0.33)	**3** (0.33)	**5** (0.56)	**3** (0.43)
LASER	**7** (0.78)	**7** (0.78)	**5** (0.56)	**8** (0.89)	**4** (0.57)

Table 36: Top-three counts (F1 scores) of each embedding in the languages considered. The numbers in parentheses represent the counts normalized by the number of probing tasks.

Language: English (Accuracy)																		
Embedding	Surface Tasks				Syntactic Tasks										Semantic Tasks			
	SentLen		WC		BiShift		SVAgree		SVDist		Voice		WO		EOS		SubjNum	
	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R	F1	R
Size of the data set																		
# instances	10,000	-	10,000	-	10,000	-	10,000	-	10,000	-	9,999	-	10,000	-	10,000	-	9,448	-
Majority class (baseline)																		
Majority	17.9	-	13.3	-	50.2	-	51.1	-	47.5	-	61.0	-	33.4	-	21.7	-	75.7	-
Random embeddings (baseline)																		
BOREP	50.728	8	65.800	10	49.780	8	79.680	7	58.659	11	79.490	7	29.591	8	27.780	6	88.073	8
Random BiLSTM	62.730	4	65.999	9	51.880	6	80.740	4	66.110	4	83.990	3	59.889	4	31.101	3	91.174	3
Average embeddings (*FastText*)																		
Vanilla average	30.220	11	75.919	8	49.760	9	77.430	9	61.680	7	80.820	6	25.811	9	24.510	10	89.703	6
p-Means	61.430	5	76.119	7	49.260	12	79.820	6	61.050	8	82.140	4	20.381	11	28.880	5	89.925	4
SIF	28.061	12	77.518	6	49.430	11	80.650	5	60.810	9	77.540	10	24.890	10	23.350	11	88.380	7
GEM	35.099	9	82.572	4	49.760	9	78.510	8	54.700	12	73.550	12	07.080	12	25.770	7	85.956	10
hier. pooling	55.801	7	55.287	12	50.700	7	65.890	11	62.810	6	78.980	9	37.451	7	30.080	4	87.089	9
Trained embeddings																		
InferSent	68.951	3	94.780	1	54.390	4	91.320	1	69.599	2	87.450	2	76.050	3	25.050	8	91.311	2
Quick-Thought	68.970	2	84.480	3	54.660	3	88.520	3	67.321	3	82.140	4	80.420	2	24.959	9	82.559	12
sent2vec	31.000	10	81.555	5	53.190	5	90.900	2	59.170	10	79.090	8	38.090	6	22.640	12	82.845	11
BERT	59.169	6	61.608	11	64.250	1	72.260	10	66.070	5	77.360	11	56.869	5	31.921	2	89.713	5
LASER	74.971	1	89.979	2	59.950	2	57.120	12	72.921	1	90.920	1	84.790	1	37.790	1	92.909	1

Table 37: Probing task results for the English language (accuracy).

C Detailed Downstream Task Results

C.1 Results measured by F1 Score

Language: English (F1)						
Embedding	**ArgMin**		**Senti**		**TREC**	
	F1	**R**	**F1**	**R**	**F1**	**R**
Size of the data set						
# instances	25,303	-	14,148	-	5,952	-
Majority class (baseline)						
Majority	56.4	-	64.0	-	22.6	-
Random embeddings (baseline)						
BOREP	00.542	13	00.646	9	00.819	5
Random BiLSTM	00.536	15	00.688	6	00.774	9
Average embeddings (*FastText*)						
Vanilla average	00.550	12	00.698	4	00.795	7
p-Means	00.568	7	00.695	5	00.767	10
SIF	00.533	16	00.702	2	00.715	13
GEM	00.540	14	00.646	9	00.820	4
hier. pooling	00.513	17	00.627	16	00.724	12
Average embeddings (*word2vec*)						
Vanilla average	00.585	4	00.639	12	00.693	16
p-Means	00.561	9	00.622	18	00.713	14
SIF	00.581	5	00.627	16	00.644	18
GEM	00.426	22	00.516	22	00.576	20
hier. pooling	00.555	11	00.603	19	00.697	15
Average embeddings (*Attract-Repel*)						
Vanilla average	00.511	18	00.630	15	00.651	17
p-Means	00.568	7	00.660	7	00.748	11
SIF	00.497	19	00.631	14	00.554	22
GEM	00.487	21	00.565	21	00.564	21
hier. pooling	00.492	20	00.594	20	00.641	19
Trained embeddings						
InferSent	00.604	2	00.702	2	00.920	1
Quick-Thought	00.581	5	00.636	13	00.814	6
sent2vec	00.608	1	00.656	8	00.792	8
BERT	00.558	10	00.644	11	00.876	2
LASER	00.596	3	00.727	1	00.851	3

Table 38: Downstream task results for the English language (F1 scores).

Language: German (F1)						
Embedding	**ArgMin**		**Senti**		**TREC**	
	F1	**R**	**F1**	**R**	**F1**	**R**
Size of the data set						
# instances	25,303	-	17,908	-	5,952	-
Majority class (baseline)						
Majority	56.4	-	67.8	-	22.6	-
Random embeddings (baseline)						
BOREP	00.507	7	00.545	8	00.764	4
Random BiLSTM	00.507	7	00.559	4	00.676	9
Average embeddings (*FastText*)						
Vanilla average	00.507	7	00.544	9	00.694	8
p-Means	00.534	2	00.550	6	00.765	3
SIF	00.514	6	00.555	5	00.734	6
GEM	00.521	4	00.572	3	00.730	7
hier. pooling	00.467	16	00.489	18	00.601	12
Average embeddings (*word2vec*)						
Vanilla average	00.501	11	00.532	11	00.581	14
p-Means	00.479	14	00.540	10	00.596	13
SIF	00.499	12	00.515	14	00.558	17
GEM	00.391	22	00.478	19	00.553	18
hier. pooling	00.471	15	00.520	13	00.569	16
Average embeddings (*Attract-Repel*)						
Vanilla average	00.432	20	00.466	20	00.488	21
p-Means	00.505	10	00.529	12	00.607	11
SIF	00.423	21	00.461	21	00.489	20
GEM	00.443	18	00.506	17	00.491	19
hier. pooling	00.434	19	00.432	22	00.408	22
Trained embeddings						
InferSent	00.456	17	00.547	7	00.580	15
Quick-Thought	00.497	13	00.511	16	00.667	10
sent2vec	00.534	2	00.590	2	00.846	1
BERT	00.517	5	00.514	15	00.802	2
LASER	00.583	1	00.623	1	00.753	5

Table 39: Downstream task results for the German language (F1 scores).

Embedding	Language: Russian (F1)					
	ArgMin		Senti		TREC	
	F1	R	F1	R	F1	R
Size of the data set						
# instances	25,303	-	30,000	-	5,952	-
Majority class (baseline)						
Majority	56.4	-	51.2	-	22.6	-
Random embeddings (baseline)						
BOREP	00.518	11	00.643	9	00.778	7
Random BiLSTM	00.521	10	00.695	4	00.739	10
Average embeddings (*FastText*)						
Vanilla average	00.535	4	00.703	2	00.767	8
p-Means	00.553	2	00.695	4	00.799	5
SIF	00.508	13	00.701	3	00.731	13
GEM	00.515	12	00.630	10	00.803	3
hier. pooling	00.486	16	00.673	7	00.717	15
Average embeddings (*word2vec*)						
Vanilla average	00.531	6	00.601	14	00.738	11
p-Means	00.524	9	00.605	13	00.801	4
SIF	00.529	8	00.598	15	00.732	12
GEM	00.386	20	00.551	22	00.546	18
hier. pooling	00.501	14	00.584	17	00.745	9
Average embeddings (*Attract-Repel*)						
Vanilla average	00.371	21	00.568	21	00.399	21
p-Means	00.478	17	00.586	16	00.551	17
SIF	00.365	22	00.582	18	00.394	22
GEM	00.433	18	00.578	19	00.405	20
hier. pooling	00.409	19	00.574	20	00.410	19
Trained embeddings						
InferSent	00.487	15	00.676	6	00.623	16
Quick-Thought	00.540	3	00.627	11	00.780	6
sent2vec	00.534	5	00.622	12	00.832	1
BERT	00.531	6	00.644	8	00.822	2
LASER	00.582	1	00.728	1	00.728	14

Table 40: Downstream task results for the Russian language (F1 scores).

Embedding	Language: Turkish (F1)					
	ArgMin		Senti		TREC	
	F1	R	F1	R	F1	R
Size of the data set						
# instances	25,303	-	6,172	-	5,952	-
Majority class (baseline)						
Majority	56.4	-	52.0	-	22.6	-
Random embeddings (baseline)						
BOREP	00.512	10	00.498	8	00.722	1
Random BiLSTM	00.515	7	00.529	1	00.685	6
Average embeddings (*FastText*)						
Vanilla average	00.525	4	00.529	1	00.649	13
p-Means	00.547	2	00.511	7	00.683	7
SIF	00.506	14	00.528	3	00.658	11
GEM	00.508	13	00.513	6	00.715	2
hier. pooling	00.509	12	00.497	9	00.626	15
Average embeddings (*word2vec*)						
Vanilla average	00.517	6	00.479	13	00.710	3
p-Means	00.513	9	00.464	19	00.683	7
SIF	00.520	5	00.477	15	00.642	14
GEM	00.386	22	00.415	21	00.546	20
hier. pooling	00.503	16	00.479	13	00.623	16
Average embeddings (*Attract-Repel*)						
Vanilla average	00.476	19	00.465	18	00.599	17
p-Means	00.540	3	00.487	10	00.660	9
SIF	00.466	20	00.457	20	00.597	18
GEM	00.435	21	00.411	22	00.397	22
hier. pooling	00.477	18	00.472	16	00.586	19
Trained embeddings						
InferSent	00.510	11	00.514	5	00.545	21
Quick-Thought	00.515	7	00.469	17	00.650	12
sent2vec	00.504	15	00.483	12	00.698	4
BERT	00.493	17	00.484	11	00.695	5
LASER	00.555	1	00.523	4	00.660	9

Table 41: Downstream task results for the Turkish language (F1 scores).

Language: Georgian (F1)						
Embedding	ArgMin		Senti		TREC	
	F1	R	F1	R	F1	R
Size of the data set						
# instances	25,303	-	11,513	-	5,952	-
Majority class (baseline)						
Majority	56.4	-	54.8	-	22.6	-
Random embeddings (baseline)						
BOREP	00.517	2	00.544	4	00.774	6
Random BiLSTM	00.506	4	00.546	3	00.770	8
Average embeddings (*FastText*)						
Vanilla average	00.500	6	00.514	7	00.739	11
p-Means	00.541	1	00.539	5	00.774	6
SIF	00.481	9	00.511	8	00.732	12
GEM	00.500	6	00.596	1	00.742	10
hier. pooling	00.460	16	00.474	14	00.747	9
Average embeddings (*word2vec*)						
Vanilla average	00.480	10	00.423	19	00.684	16
p-Means	00.480	10	00.568	2	00.796	3
SIF	00.473	14	00.406	21	00.671	19
GEM	00.388	22	00.477	13	00.673	18
hier. pooling	00.475	13	00.465	15	00.723	14
Average embeddings (*Attract-Repel*)						
Vanilla average	00.460	16	00.401	22	00.705	15
p-Means	00.512	3	00.484	12	00.841	1
SIF	00.429	21	00.418	20	00.730	13
GEM	00.434	20	00.443	17	00.619	20
hier. pooling	00.450	18	00.432	18	00.684	16
Trained embeddings						
InferSent	00.479	12	00.510	10	00.578	21
Quick-Thought	00.482	8	00.488	11	00.811	2
sent2vec	00.472	15	00.531	6	00.787	4
BERT	00.502	5	00.511	8	00.784	5
LASER	00.449	19	00.450	16	00.523	22

Table 42: Downstream task results for the Georgian language (F1 scores).

C.2 Results measured by Accuracy

Language: English (Accuracy)						
Embedding	ArgMin		Senti		TREC	
	F1	R	F1	R	F1	R
Size of the data set						
# instances	25,303	-	14,148	-	5,952	-
Majority class (baseline)						
Majority	56.4	-	64.0	-	22.6	-
Random embeddings (baseline)						
BOREP	63.030	10	74.366	7	84.500	4
Random BiLSTM	63.282	9	77.200	5	79.600	7
Average embeddings (*FastText*)						
Vanilla average	65.191	5	78.380	3	79.600	7
p-Means	64.709	6	77.695	4	81.800	5
SIF	63.516	8	78.521	2	76.200	11
GEM	61.484	12	72.267	12	78.800	9
hier. pooling	62.385	11	74.288	8	76.000	12
Trained embeddings						
InferSent	67.321	3	77.157	6	91.400	1
Quick-Thought	66.235	4	73.016	10	81.400	6
sent2vec	67.693	2	74.125	9	78.200	10
BERT	63.642	7	73.002	11	90.800	2
LASER	67.807	1	79.942	1	90.200	3

Table 43: Downstream task results for the English language (accuracy).

Language: German (Accuracy)						
Embedding	ArgMin		Senti		TREC	
	F1	R	F1	R	F1	R
Size of the data set						
# instances	25,303	-	14,148	-	5,952	-
Majority class (baseline)						
Majority	56.4	-	64.0	-	22.6	-
Random embeddings (baseline)						
BOREP	60.273	10	71.283	8	80.400	4
Random BiLSTM	61.813	5	72.477	7	73.600	9
Average embeddings (*FastText*)						
Vanilla average	63.066	3	73.924	3	75.200	6
p-Means	64.287	2	73.282	4	75.000	7
SIF	62.546	4	74.008	2	75.600	5
GEM	60.056	11	70.568	10	74.000	8
hier. pooling	60.777	8	70.741	9	71.800	10
Trained embeddings						
InferSent	60.808	7	72.595	6	71.200	11
Quick-Thought	58.709	12	67.781	12	68.200	12
sent2vec	61.131	6	72.723	5	84.200	2
BERT	60.379	9	68.831	11	81.400	3
LASER	67.179	1	75.833	1	85.200	1

Table 44: Downstream task results for the German language (accuracy).

D Detailed Stability Analysis Results

D.1 English

Ø 5,000	LR		NN		NN_H		RF		SVM		NB	
	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank
Vanilla Average	0,40	7	0,41	9	0,42	8	0,22	9	0,44	6	0,30	7
GEM	0,68	4	0,69	4	0,67	4	0,25	7	0,65	5	0,43	4
SIF	0,34	9	0,45	5	0,44	7	0,25	6	0,41	8	0,36	5
Power Means	0,36	8	0,44	7	0,48	6	0,27	5	0,37	10	0,16	9
Hierarchical Pooling	0,33	10	0,21	12	0,24	12	0,16	12	0,35	11	0,15	10
BOREP	0,30	11	0,36	10	0,40	10	0,39	4	0,38	9	0,11	12
Random BiLSTM	0,16	12	0,26	11	0,30	11	0,20	10	0,16	12	0,12	11
sent2vec	0,72	2	0,73	2	0,69	3	0,24	8	0,67	4	0,56	1
Quick-Thought	0,68	3	0,72	3	0,72	2	0,41	3	0,71	2	0,48	2
LASER	0,56	5	0,45	6	0,66	5	0,55	2	0,70	3	0,44	3
BERT	0,43	6	0,44	8	0,41	9	0,17	11	0,42	7	0,25	8
InferSent	0,79	1	0,78	1	0,87	1	0,60	1	0,81	1	0,34	6

Table 45: Stability analysis results for 5k instances (WC task, EN).

Ø 10,000	LR		NN		NN_H		RF		SVM		NB	
	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank
Vanilla Average	0,50	7	0,54	8	0,58	8	0,25	9	0,53	6	0,35	7
GEM	0,71	4	0,72	4	0,68	5	0,28	7	0,71	5	0,43	4
SIF	0,45	8	0,57	6	0,59	7	0,30	6	0,51	7	0,40	5
Power Means	0,41	9	0,56	7	0,60	6	0,34	5	0,46	9	0,20	9
Hierarchical Pooling	0,38	10	0,32	12	0,34	12	0,19	11	0,42	11	0,19	10
BOREP	0,34	11	0,46	10	0,49	9	0,45	4	0,46	8	0,12	12
Random BiLSTM	0,21	12	0,40	11	0,48	11	0,23	10	0,24	12	0,14	11
sent2vec	0,77	2	0,79	3	0,76	4	0,27	8	0,75	4	0,64	1
Quick-Thought	0,74	3	0,81	2	0,80	3	0,47	3	0,77	3	0,50	2
LASER	0,69	5	0,65	5	0,81	2	0,60	2	0,79	2	0,48	3
BERT	0,51	6	0,48	9	0,49	10	0,18	12	0,44	10	0,25	8
InferSent	0,87	1	0,87	1	0,89	1	0,66	1	0,86	1	0,40	6

Table 46: Stability analysis results for 10k instances (WC task, EN).

Ø 30,000	LR		NN		NN_H		RF		SVM		NB	
	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank
Vanilla Average	0,62	6	0,72	8	0,75	7	0,28	9	--	--	0,38	7
GEM	0,77	5	0,78	5	0,74	8	0,33	8	--	--	0,48	5
SIF	0,60	7	0,75	6	0,76	6	0,34	7	--	--	0,41	6
Power Means	0,45	9	0,73	7	0,76	5	0,41	5	--	--	0,24	9
Hierarchical Pooling	0,44	10	0,47	12	0,48	12	0,22	11	--	--	0,22	10
BOREP	0,39	11	0,59	9	0,62	10	0,55	3	--	--	0,14	12
Random BiLSTM	0,31	12	0,59	10	0,67	9	0,26	10	--	--	0,15	11
sent2vec	0,80	4	0,80	4	0,79	4	0,35	6	--	--	0,63	1
Quick-Thought	0,83	2	0,87	2	0,88	3	0,53	4	--	--	0,50	4
LASER	0,82	3	0,86	3	0,90	2	0,67	2	--	--	0,54	2
BERT	0,55	8	0,58	11	0,54	11	0,22	12	--	--	0,28	8
InferSent	0,93	1	0,94	1	0,95	1	0,72	1	--	--	0,51	3

Table 47: Stability analysis results for 30k instances (WC task, EN).

Ø 60,000	LR		NN		NN_H		RF		SVM		NB	
	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank
Vanilla Average	0,66	6	0,78	8	0,81	6	0,31	9	--	--	0,38	7
GEM	0,81	5	0,81	5	0,78	8	0,36	8	--	--	0,48	5
SIF	0,65	7	0,79	7	0,81	7	0,38	7	--	--	0,43	6
Power Means	0,45	10	0,80	6	0,82	4	0,45	5	--	--	0,23	9
Hierarchical Pooling	0,45	9	0,51	12	0,52	12	0,24	11	--	--	0,21	10
BOREP	0,40	11	0,66	10	0,67	10	0,59	3	--	--	0,15	12
Random BiLSTM	0,36	12	0,68	9	0,75	9	0,30	10	--	--	0,16	11
sent2vec	0,83	4	0,82	4	0,81	5	0,39	6	--	--	0,64	1
Quick-Thought	0,86	3	0,92	2	0,91	3	0,57	4	--	--	0,50	4
LASER	0,88	2	0,91	3	0,92	2	0,72	2	--	--	0,55	3
BERT	0,57	8	0,59	11	0,57	11	0,22	12	--	--	0,27	8
InferSent	0,96	1	0,97	1	0,98	1	0,77	1	--	--	0,57	2

Table 48: Stability analysis results for 60k instances (WC task, EN).

5,000 instances

	LR	NN	NN_H	RF	SVM	NB
LR	1,00					
NN	0,89	1,00				
NN_H	0,91	0,97	1,00			
RF	0,45	0,61	0,69	1,00		
SVM	0,92	0,84	0,88	0,63	1,00	
NB	0,81	0,82	0,80	0,36	0,80	1,00

Figure 31: Classifier correlations with 5k instances (WC task, EN).

10,000 instances

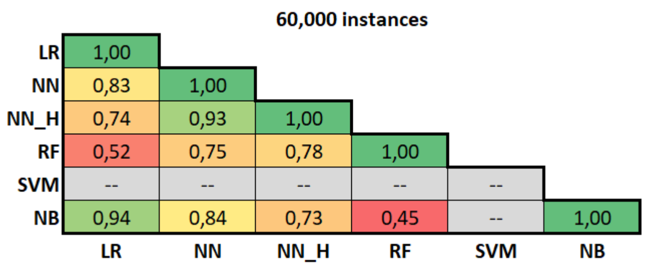
	LR	NN	NN_H	RF	SVM	NB
LR	1,00					
NN	0,91	1,00				
NN_H	0,83	0,94	1,00			
RF	0,41	0,66	0,81	1,00		
SVM	0,85	0,90	0,94	0,76	1,00	
NB	0,84	0,83	0,76	0,34	0,78	1,00

Figure 32: Classifier correlations with 10k instances (WC task, EN).

30,000 instances

	LR	NN	NN_H	RF	SVM	NB
LR	1,00					
NN	0,90	1,00				
NN_H	0,82	0,94	1,00			
RF	0,55	0,78	0,78	1,00		
SVM	--	--	--	--	--	
NB	0,92	0,84	0,79	0,45	--	1,00

Figure 33: Classifier correlations with 30k instances (WC task, EN).



60,000 instances

	LR	NN	NN_H	RF	SVM	NB
LR	1,00					
NN	0,83	1,00				
NN_H	0,74	0,93	1,00			
RF	0,52	0,75	0,78	1,00		
SVM	--	--	--	--	--	
NB	0,94	0,84	0,73	0,45	--	1,00

Figure 34: Classifier correlations with 60k instances (WC task, EN).

Absolute performance deltas in the WordContent task (NN_H classifier)											
Embedding	❶ Δ (im-)balance			❷ Δ (no) HP tuning			❸ Δ NN ⟷ NN_H			❹ Δ size	
	10k	30k	60k	10k	30k	60k	10k	30k	60k	10k ⟷ 30k	30k ⟷ 60k
Vanilla average	0.21	0.13	0.09	0.05	0.01	0.00	-0.04	-0.04	-0.02	0.09	0.02
p-Means	0.20	0.12	0.09	0.02	0.00	0.00	-0.03	-0.02	-0.01	0.08	0.03
SIF	0.24	0.14	0.10	0.05	0.00	0.01	-0.01	-0.02	-0.01	0.07	0.01
GEM	0.17	0.12	0.09	0.03	0.04	0.04	0.01	0.02	0.03	0.01	0.01
hier. pooling	0.29	0.23	0.22	0.03	0.01	0.00	-0.04	-0.02	-0.01	0.08	0.03
BOREP	0.25	0.20	0.18	0.01	0.00	0.00	-0.01	-0.02	0.00	0.08	0.03
Random BiLSTM	0.23	0.17	0.14	0.05	0.00	0.00	-0.08	-0.07	-0.06	0.13	0.05
InferSent	0.07	0.03	0.01	0.01	0.00	0.00	-0.02	0.00	0.00	0.02	0.01
Quick-Thought	0.06	0.03	0.02	0.00	0.00	0.01	-0.02	0.00	0.00	0.05	0.02
sent2vec	0.22	0.20	0.18	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00
BERT	0.05	0.04	0.04	0.01	0.02	0.03	0.02	0.03	0.02	0.04	0.03
LASER	0.09	0.03	0.02	0.02	0.01	0.00	-0.08	-0.02	-0.01	0.03	0.01

Table 49: Absolute F1 performance deltas in the WC task (`NN_H`): ❶ Effect of `class balance` (positive number indicates positive effect of balanced data), ❷ Effect of `hyper-parameter tuning` (on balanced data), ❸ Effect of `classifier` (on balanced data; negative values indicate worse performance of `NN`) and ❹ effect of `size` (on balanced data sets).

Effect of hyper-parameter tuning in the WordContent task (NN_H classifier)													
Embedding	Dim.	10k				30k				60k			
		imbal.	R	bal.	R	imbal.	R	bal.	R	imbal.	R	bal.	R
Vanilla average	300	0.70	8	0.84	7	-	-	0.89	7	-	-	0.90	8
p-Means	1,500	0.74	7	0.82	8	-	-	0.88	8	-	-	0.91	6
SIF	300	0.79	6	0.88	4	-	-	0.90	5	-	-	0.92	5
GEM	300	0.84	4	0.88	4	-	-	0.90	5	-	-	0.91	6
hier. pooling	300	0.48	12	0.66	11	-	-	0.72	11	-	-	0.74	11
BOREP	4,096	0.53	11	0.75	10	-	-	0.82	10	-	-	0.85	10
Random BiLSTM	8,129	0.63	9	0.76	9	-	-	0.84	9	-	-	0.89	9
InferSent	4,096	0.93	1	0.97	2	-	-	0.98	2	-	-	0.99	1
Quick-Thought	2,400	0.87	3	0.86	6	-	-	0.91	4	-	-	0.94	3
sent2vec	700	0.80	5	0.99	1	-	-	0.99	1	-	-	0.99	1
BERT	768	0.55	10	0.55	12	-	-	0.60	12	-	-	0.64	12
LASER	1,024	0.90	2	0.92	3	-	-	0.94	3	-	-	0.94	3

Table 50: Effect of hyper-parameter tuning on the WC probing task results (F1 scores). The results are reported for the `NN_H` classifier.

Effects of data set size and class balance in the WordContent task													
Embedding	Dim.	10k				30k				60k			
		imbal.	R	bal.	R	imbal.	R	bal.	R	imbal.	R	bal.	R
Neural network with hidden layer (`NN_H`)													
Vanilla average	300	0.58	8	0.79	8	0.75	7	0.88	6	0.81	5	0.90	7
p-Means	1,500	0.60	6	0.80	7	0.76	5	0.88	6	0.82	4	0.91	5
SIF	300	0.59	7	0.83	6	0.76	5	0.90	5	0.81	5	0.91	5
GEM	300	0.68	5	0.85	5	0.74	8	0.86	8	0.78	8	0.87	9
hier. pooling	300	0.34	12	0.63	11	0.48	12	0.71	11	0.52	12	0.74	11
BOREP	4,096	0.49	9	0.74	9	0.62	10	0.82	10	0.67	10	0.85	10
Random BiLSTM	8,129	0.48	11	0.71	10	0.67	9	0.84	9	0.75	9	0.89	8
InferSent	4,096	0.89	1	0.96	2	0.95	1	0.98	2	0.98	1	0.99	1
Quick-Thought	2,400	0.80	3	0.86	4	0.88	3	0.91	4	0.91	3	0.93	4
sent2vec	700	0.76	4	0.98	1	0.79	4	0.99	1	0.81	5	0.99	1
BERT	768	0.49	9	0.54	12	0.54	11	0.58	12	0.57	11	0.61	12
LASER	1,024	0.81	2	0.90	3	0.90	2	0.93	3	0.92	2	0.94	3
Neural net without hidden layer (`NN`)													
Vanilla average	300	0.54	8	0.75	8	0.72	8	0.84	8	0.78	8	0.88	8
p-Means	1,500	0.56	7	0.77	7	0.73	7	0.86	7	0.80	6	0.90	5
SIF	300	0.57	6	0.82	5	0.75	6	0.88	5	0.79	7	0.90	5
GEM	300	0.72	4	0.86	3	0.78	5	0.88	5	0.81	5	0.90	5
hier. pooling	300	0.32	12	0.59	11	0.47	12	0.69	11	0.51	12	0.73	11
BOREP	4,096	0.46	10	0.73	9	0.59	9	0.80	9	0.66	10	0.85	9
Random BiLSTM	8,129	0.40	11	0.63	10	0.59	9	0.77	10	0.68	9	0.83	10
InferSent	4,096	0.87	1	0.94	2	0.94	1	0.98	2	0.97	1	0.99	1
Quick-Thought	2,400	0.81	2	0.84	4	0.87	2	0.91	3	0.92	2	0.93	3
sent2vec	700	0.79	3	0.98	1	0.80	4	0.99	1	0.82	4	0.99	1
BERT	768	0.48	9	0.56	12	0.58	11	0.61	12	0.59	11	0.63	12
LASER	1,024	0.65	5	0.82	5	0.86	3	0.91	3	0.91	3	0.93	3

Table 51: Effects of class (im-)balance as well as data set size on the results for the WC probing task (F1 scores). The results are reported for the `NN_H` classifier (top) as well as for the `NN` classifier (bottom).

Ø 5,000	LR		NN		NN_H		RF		SVM		NB	
	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank
Vanilla Average	0,43	7	0,49	5	0,49	6	0,32	6	0,45	6	0,47	2
GEM	0,77	2	0,77	2	0,74	2	0,27	8	0,76	1	0,43	4
SIF	0,37	9	0,45	8	0,48	7	0,32	7	0,39	9	0,46	3
Power Means	0,50	4	0,56	4	0,55	4	0,44	3	0,53	4	0,28	6
Hierarchical Pooling	0,38	8	0,33	10	0,34	10	0,22	11	0,43	7	0,25	7
BOREP	0,49	5	0,47	6	0,51	5	0,51	1	0,52	5	0,21	9
Random BiLSTM	0,17	11	0,38	9	0,40	9	0,36	4	0,16	11	0,21	11
sent2vec	0,78	1	0,78	1	0,76	1	0,46	2	0,73	2	0,56	1
Quick-Thought	0,55	3	0,61	3	0,63	3	0,34	5	0,62	3	0,28	5
LASER	0,19	10	0,21	11	0,25	12	0,24	10	0,17	10	0,22	8
BERT	0,45	6	0,45	7	0,42	8	0,21	12	0,42	8	0,21	10
InferSent	0,12	12	0,19	12	0,26	11	0,26	9	0,10	12	0,13	12

Table 52: Stability analysis results for 5k instances (WC task, KA).

Ø 10,000	LR		NN		NN_H		RF		SVM		NB	
	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank
Vanilla Average	0,50	6	0,65	5	0,64	5	0,35	6	0,54	6	0,47	2
GEM	0,79	1	0,80	1	0,77	2	0,31	8	0,80	1	0,41	4
SIF	0,46	8	0,62	6	0,60	7	0,34	7	0,49	8	0,46	3
Power Means	0,57	5	0,67	4	0,69	4	0,49	2	0,60	5	0,27	6
Hierarchical Pooling	0,46	7	0,43	10	0,46	10	0,26	11	0,49	7	0,24	8
BOREP	0,58	4	0,61	7	0,62	6	0,57	1	0,61	4	0,21	11
Random BiLSTM	0,23	11	0,49	9	0,53	8	0,41	4	0,26	10	0,22	10
sent2vec	0,77	2	0,75	2	0,79	1	0,47	3	0,72	2	0,52	1
Quick-Thought	0,66	3	0,69	3	0,71	3	0,39	5	0,66	3	0,28	5
LASER	0,24	10	0,30	12	0,35	12	0,29	10	0,23	11	0,22	9
BERT	0,46	9	0,50	8	0,47	9	0,25	12	0,46	9	0,24	7
InferSent	0,15	12	0,31	11	0,39	11	0,31	9	0,15	12	0,10	12

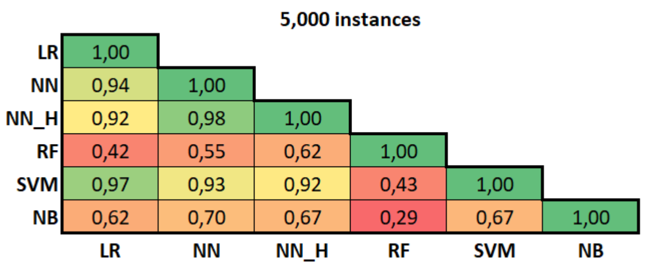
Table 53: Stability analysis results for 10k instances (WC task, KA).

Ø 30,000	LR		NN		NN_H		RF		SVM		NB	
	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank
Vanilla Average	0,62	5	0,79	4	0,79	4	0,41	7	--	--	0,47	3
GEM	0,82	2	0,83	2	0,80	3	0,37	9	--	--	0,43	4
SIF	0,57	7	0,77	5	0,77	5	0,41	6	--	--	0,47	2
Power Means	0,64	4	0,79	3	0,82	2	0,57	2	--	--	0,29	5
Hierarchical Pooling	0,54	8	0,60	9	0,60	9	0,32	10	--	--	0,26	7
BOREP	0,62	6	0,72	7	0,73	7	0,64	1	--	--	0,21	11
Random BiLSTM	0,33	10	0,66	8	0,72	8	0,47	5	--	--	0,22	10
sent2vec	0,85	1	0,84	1	0,83	1	0,53	3	--	--	0,51	1
Quick-Thought	0,73	3	0,76	6	0,75	6	0,47	4	--	--	0,28	6
LASER	0,31	11	0,42	12	0,46	12	0,32	11	--	--	0,23	9
BERT	0,49	9	0,57	10	0,56	11	0,28	12	--	--	0,23	8
InferSent	0,24	12	0,50	11	0,59	10	0,38	8	--	--	0,09	12

Table 54: Stability analysis results for 30k instances (WC task, KA).

Ø 60,000	LR		NN		NN_H		RF		SVM		NB	
	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank
Vanilla Average	0,69	4	0,84	3	0,84	3	0,43	7	--	--	0,47	2
GEM	0,84	2	0,84	4	0,81	5	0,41	9	--	--	0,43	4
SIF	0,63	7	0,83	5	0,82	4	0,43	6	--	--	0,46	3
Power Means	0,66	5	0,85	2	0,87	1	0,60	2	--	--	0,29	6
Hierarchical Pooling	0,58	8	0,67	9	0,66	10	0,36	10	--	--	0,28	7
BOREP	0,64	6	0,79	7	0,80	6	0,68	1	--	--	0,22	11
Random BiLSTM	0,37	10	0,75	8	0,80	7	0,49	5	--	--	0,23	10
sent2vec	0,87	1	0,87	1	0,86	2	0,55	3	--	--	0,51	1
Quick-Thought	0,76	3	0,81	6	0,80	8	0,51	4	--	--	0,30	5
LASER	0,34	11	0,50	12	0,54	12	0,35	11	--	--	0,24	9
BERT	0,51	9	0,62	10	0,60	11	0,30	12	--	--	0,25	8
InferSent	0,28	12	0,61	11	0,67	9	0,41	8	--	--	0,09	12

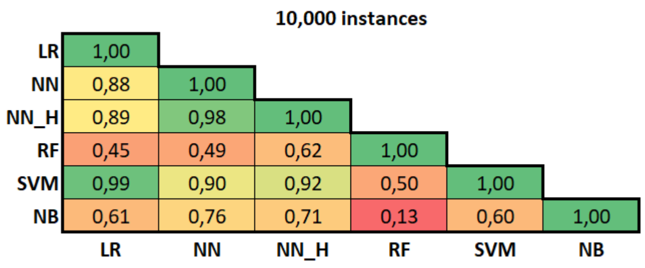
Table 55: Stability analysis results for 60k instances (WC task, KA).



5,000 instances

	LR	NN	NN_H	RF	SVM	NB
LR	1,00					
NN	0,94	1,00				
NN_H	0,92	0,98	1,00			
RF	0,42	0,55	0,62	1,00		
SVM	0,97	0,93	0,92	0,43	1,00	
NB	0,62	0,70	0,67	0,29	0,67	1,00

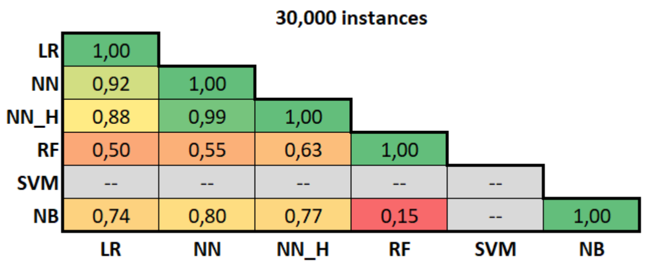
Figure 35: Classifier correlations with 5k instances (WC task, KA).



10,000 instances

	LR	NN	NN_H	RF	SVM	NB
LR	1,00					
NN	0,88	1,00				
NN_H	0,89	0,98	1,00			
RF	0,45	0,49	0,62	1,00		
SVM	0,99	0,90	0,92	0,50	1,00	
NB	0,61	0,76	0,71	0,13	0,60	1,00

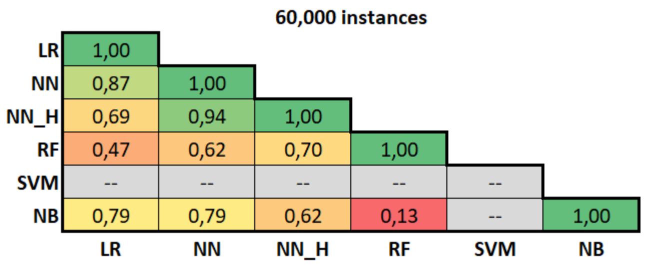
Figure 36: Classifier correlations with 10k instances (WC task, KA).



30,000 instances

	LR	NN	NN_H	RF	SVM	NB
LR	1,00					
NN	0,92	1,00				
NN_H	0,88	0,99	1,00			
RF	0,50	0,55	0,63	1,00		
SVM	--	--	--	--	--	
NB	0,74	0,80	0,77	0,15	--	1,00

Figure 37: Classifier correlations with 30k instances (WC task, KA).



60,000 instances

	LR	NN	NN_H	RF	SVM	NB
LR	1,00					
NN	0,87	1,00				
NN_H	0,69	0,94	1,00			
RF	0,47	0,62	0,70	1,00		
SVM	--	--	--	--	--	
NB	0,79	0,79	0,62	0,13	--	1,00

Figure 38: Classifier correlations with 60k instances (WC task, KA).

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi and Yoav Goldberg (2017). Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *CoRR*, abs/1608.04207. URL https://arxiv.org/abs/1608.04207.

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow and Rebecca Passonneau (2011). Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, LSM 2011, pages 30–38. Association for Computational Linguistics, Stroudsburg, PA, USA. ISBN 978-1-932432-96-1. URL http://dl.acm.org/citation.cfm?id=2021109.2021114.

Zeljko Agic and Natalie Schluter (2017). Baselines and test data for cross-lingual inference. *CoRR*, abs/1704.05347. URL http://arxiv.org/abs/1704.05347.

Lars Ahrenberg (2007). LinES: An English-Swedish parallel treebank. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007)*, pages 270–273. University of Tartu, Tartu, Estonia. URL https://www.aclweb.org/anthology/W07-2441.

Lars Ahrenberg (2015). Converting an English-Swedish parallel treebank to universal dependencies. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 10–19. Uppsala University, Uppsala, Sweden. URL https://www.aclweb.org/anthology/W15-2103.

Mansur Alp Tocoglu and Adil Alpkocak (2018). TREMO: A dataset for emotion analysis in Turkish. *Journal of Information Science*, 44. doi:10.1177/0165551518761014. URL https://journals.sagepub.com/doi/abs/10.1177/0165551518761014.

Sanjeev Arora, Yingyu Liang and Tengyu Ma (2017). A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. URL https://openreview.net/forum?id=SyK00v5xx.

Mikel Artetxe, Gorka Labaka and Eneko Agirre (2016). Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294. Association for Computational Linguistics, Austin, Texas. doi:10.18653/v1/D16-1250. URL https://www.aclweb.org/anthology/D16-1250.

Mikel Artetxe and Holger Schwenk (2018). Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *CoRR*, abs/1812.10464. URL https://arxiv.org/abs/1812.10464.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent and Christian Janvin (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944966.

Yevgeni Berzak, Jessica Kenney, Carolyn Spadine, Jing Xian Wang, Lucia Lam, Keiko Sophie Mori, Sebastian Garza and Boris Katz (2016). Universal dependencies for learner English. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 737–746. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1070.

Steven Bird and Edward Loper (2002). NLTK: The natural language toolkit. *CoRR*, cs.CL/0205028. URL https://arxiv.org/abs/cs/0205028.

Kathryn Bock and Carol Ann Miller (1991). Broken agreement. *Cognitive Psychology*, 23:45–93. URL https://pennstate.pure.elsevier.com/en/publications/broken-agreement.

Piotr Bojanowski, Edouard Grave, Armand Joulin and Tomas Mikolov (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5(1):135–146. URL https://www.aclweb.org/anthology/Q17-1010.

Filip Boltužić and Jan Šnajder (2014). Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58. Association for Computational Linguistics, Baltimore, Maryland. doi:10.3115/v1/W14-2107. URL https://www.aclweb.org/anthology/W14-2107.

Samuel Bowman, Gabor Angeli, Christopher Potts and Christopher Manning (**2015**). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics, Lisbon, Portugal. doi:10.18653/v1/D15-1075. URL https://www.aclweb.org/anthology/D15-1075.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith and Hans Uszkoreit (**2004**). TIGER: Linguistic interpretation of a German corpus. *Journal of Language and Computation*, 2:597–620. doi:10.1007/s11168-004-7431-3. URL https://www.researchgate.net/publication/226210609_TIGER_Linguistic_interpretation_of_a_German_corpus.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio and Lucia Specia (**2017**). SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14. Vancouver, Canada. URL http://nlp.arizona.edu/SemEval-2017/pdf/SemEval001.pdf.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope and Ray Kurzweil (**2018**). Universal sentence encoder. *CoRR*, abs/1803.11175. URL http://arxiv.org/abs/1803.11175.

Sarath Chandar, Mitesh Khapra, Balaraman Ravindran, Vikas Raykar and Amrita Saha (**2013**). Multilingual deep learning. URL http://sarathchandar.in/paper/MultilingualDeepLearning.pdf.

Tea Chapidze-Morgenroth (**2015**). Sprachenstreckbrief Georgisch. URL http://www.schule-mehrsprachig.at/fileadmin/schule_mehrsprachig/redaktion/sprachensteckbriefe/SSB_2019/georgisch.pdf.

Billy Chiu, Anna Korhonen and Sampo Pyysalo (**2016**). Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 1–6. Association for Computational Linguistics, Berlin, Germany. doi:10.18653/v1/W16-2501. URL https://www.aclweb.org/anthology/W16-2501.

Nurendra Choudhary, Rajat Singh, Vijjini Anvesh Rao and Manish Shrivastava (**2018**). Twitter corpus of resource-scarce languages for sentiment analysis and multilingual emoji prediction. In *COLING*, pages 1570–1577. Association for Computational Linguistics. ISBN 978-1-948087-50-6. URL http://dblp.uni-trier.de/db/conf/coling/coling2018.html#ChoudharySRS18.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho and Yoshua Bengio (**2014**). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555. URL http://arxiv.org/abs/1412.3555.

Mark Cieliebak, Jan Milan Deriu, Dominic Egger and Fatih Uzdilli (**2017**). A twitter corpus and benchmark resources for German sentiment analysis. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 45–51. Association for Computational Linguistics, Valencia, Spain. doi:10.18653/v1/W17-1106. URL https://www.aclweb.org/anthology/W17-1106.

Cağrı Cöltekin (**2017**). A grammar-book treebank of Turkish. In *Proceedings of the 14th Workshop on Treebanks and Linguistic Theories (TLT 2014)*. URL http://coltekin.net/cagri/papers/coltekin2015tlt.pdf.

Alexis Conneau and Douwe Kiela (**2018**). SentEval: An evaluation toolkit for universal sentence representations. *CoRR*, abs/1803.05449. URL http://arxiv.org/abs/1803.05449.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault and Antoine Bordes (**2017**). Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680. Association for Computational Linguistics, Copenhagen, Denmark. doi:10.18653/v1/D17-1070. URL https://www.aclweb.org/anthology/D17-1070.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault and Marco Baroni (**2018**a). What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *CoRR*, abs/1805.01070. URL https://arxiv.org/abs/1805.01070.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel Bowman, Holger Schwenk and Veselin Stoyanov (**2018**b). XNLI: Evaluating cross-lingual sentence representations. *CoRR*, abs/1809.05053. URL https://arxiv.org/abs/1809.05053.

Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova (**2018**). BERT: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805. URL https://arxiv.org/abs/1810.04805.

Kira Droganova, Olga Lyashevskaya and Daniel Zeman (**2018**). Data conversion and consistency of monolingual corpora: Russian UD treebanks. In *Proceedings of the 17th International Workshop on Treebanks and Linguistic Theories (TLT 2018)*, pages 53–66. Linköping University Electronic Press, Linköping, Sweden. ISBN 978-91-7685-137-1. URL http://www.ep.liu.se/ecp/155/007/ecp18155007.pdf.

John Duchi, Elad Hazan and Yoram Singer (**2011**). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159. URL https://dl.acm.org/citation.cfm?id=2021068.

Steffen Eger, Andreas Rücklé and Iryna Gurevych (**2019**). Pitfalls in the evaluation of sentence embeddings. *CoRR*, abs/1906.01575. URL http://arxiv.org/abs/1906.01575.

Cristina España-Bonet, Ádám Csaba Varga, Alberto Barrón-Cedeño and Josef van Genabith (**2017**). An empirical analysis of nmt-derived interlingual embeddings and their use in parallel sentence identification. *CoRR*, abs/1704.05415. URL https://arxiv.org/abs/1704.05415v2.

Allyson Ettinger, Ahmed Elgohary and Philip Resnik (**2016**). Probing for semantic evidence of composition by means of simple classification tasks. In *RepEval@ACL*, pages 134–139. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W16-2524/.

Ludwig Fahrmeir, Rita Kuenstler, Iris Pigeot and Gerhard Tutz (**2011**). *Statistik: Der Weg zur Datenanalyse*. Springer, Berlin, 7. Auflage, korrigierter Nachdruck edition. ISBN 9783642019388. URL https://www.springer.com/de/book/9783540697398.

Xing Fang and Justin Zhan (**2015**). Sentiment analysis using product review data. *Journal of Big Data*, 2(1):5. ISSN 2196-1115. doi:10.1186/s40537-015-0015-2. URL https://doi.org/10.1186/s40537-015-0015-2.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy and Noah A. Smith (**2015**). Retrofitting word vectors to semantic lexicons. *CoRR*, abs/1411.4166. URL http://dblp.uni-trier.de/db/journals/corr/corr1411.html#FaruquiDJDHS14.

Kilian Foth, Arne Köhn, Niels Beuck and Wolfgang Menzel (**2014**). Because size does matter: The Hamburg dependency treebank. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 2326–2333. European Languages Resources Association (ELRA), Reykjavik, Iceland. URL http://www.lrec-conf.org/proceedings/lrec2014/pdf/860_Paper.pdf.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen and Marco Baroni (**2018**). Colorless green recurrent networks dream hierarchically. *CoRR*, abs/1803.11138. URL https://arxiv.org/abs/1803.11138.

Ivan Habernal, Judith Eckle-Kohler and Iryna Gurevych (**2014**). Argumentation mining on the web from information seeking perspective. In *Proceedings of the Workshop on Frontiers and Connections between Argumentation Theory and Natural Language Processing, Forlì-Cesena, Italy, July 21-25, 2014*. URL http://ceur-ws.org/Vol-1341/paper4.pdf.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield and Johnathan Weese (**2013**). UMBC_EBIQUITY-CORE: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics. URL https://ebiquity.umbc.edu/_file_directory_/papers/674.pdf.

Ahmet Hayran and Mustafa Sert (**2017**). Sentiment analysis on microblog data based on word embedding and fusion techniques. In *2017 25th Signal Processing and Communication Application Conference (SIU)*, pages 1–4. doi:10.1109/SIU.2017.7960519. URL https://ieeexplore.ieee.org/document/7960519.

Erhard Hinrichs, Julia Bartels, Yasuhiro Kawata, Valia Kordoni and Heike Telljohann (**2000**). *The Tübingen treebanks for spoken German, English and Japanese*, pages 550–574. ISBN 978-3-642-08730-1. doi:10.1007/978-3-662-04230-4_40. URL https://link.springer.com/chapter/10.1007/978-3-662-04230-4_40.

Sepp Hochreiter (**1998**). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116. URL http://dblp.uni-trier.de/db/journals/ijufks/ijufks6.html#Hochreiter98.

Sepp Hochreiter and Jürgen Schmidhuber (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780. ISSN 0899-7667. doi:10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Herbert Jaeger (2001). The 'echo state' approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148. URL https://www.researchgate.net/publication/215385037_The_echo_state_approach_to_analysing_and_training_recurrent_neural_networks-with_an_erratum_note'.

Mümtaz Karakurt (2006). Sprachenstreckbrief Türkisch. URL http://www.schule-mehrsprachig.at/fileadmin/schule_mehrsprachig/redaktion/sprachensteckbriefe/SSB_2019/tuerkisch.pdf.

Najoung Kim, Roma Patel, Adam Poliak, Alex Wang, Patrick Xia, Thomas McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, Samuel Bowman and Ellie Pavlick (2019). Probing what different NLP tasks teach machines about function word comprehension. doi:10.18653/v1/S19-1026. URL https://www.aclweb.org/anthology/S19-1026/.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard Zemel, Antonio Torralba, Raquel Urtasun and Sanja Fidler (2015). Skip-thought vectors. *CoRR*, abs/1506.06726. URL http://arxiv.org/abs/1506.06726.

Matthew Koehler, Spencer Greenhalgh and Andrea Zellner (2015). Potential applications of sentiment analysis in educational research and practice – Is SITE the friendliest conference? In *Proceedings of Society for Information Technology and Teacher Education International Conference 2015*, pages 1348–1354. Association for the Advancement of Computing in Education (AACE), Las Vegas, NV, United States. URL https://www.learntechlib.org/p/150179.

Katarzyna Krasnowska-Kieraś and Alina Wróblewska (2019). Empirical linguistic study of sentence embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5729–5739. Association for Computational Linguistics, Florence, Italy. URL https://www.aclweb.org/anthology/P19-1573.

Alice Lai and Julia Hockenmaier (2014). Illinois-LH: A denotational and distributional approach to semantics. In *SemEval@COLING*, pages 329–334. The Association for Computer Linguistics. doi:10.3115/v1/S14-2055.

Quoc Le and Tomas Mikolov (2014). Distributed representations of sentences and documents. *CoRR*, abs/1405.4053. URL http://arxiv.org/abs/1405.4053.

Ji-Ung Lee, Steffen Eger, Johannes Daxenberger and Iryna Gurevych (2017). UKP TU-DA at GermEval 2017: Deep learning for aspect based sentiment detection. In *Proceedings of the GermEval 2017*. URL https://download.hrz.tu-darmstadt.de/media/FB20/Dekanat/Publikationen/UKP/2017_GSCL_GermEval_Workshop_SharedTask.pdf.

Xin Li and Dan Roth (2002). Learning Question Classifiers. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 556–562. URL http://cogcomp.org/papers/qc-coling02.pdf.

Xin Li and Dan Roth (2005). Learning Question Classifiers: The Role of Semantic Information. *Journal of Natural Language Engineering*, 11(4). URL http://cogcomp.org/papers/LiRo05a.pdf.

Tal Linzen, Emmanuel Dupoux and Yoav Goldberg (2016). Assessing the ability of lstms to learn syntax-sensitive dependencies. *CoRR*, abs/1611.01368. URL http://arxiv.org/abs/1611.01368.

Lajanugen Logeswaran and Honglak Lee (2018). An efficient framework for learning sentence representations. *CoRR*, abs/1803.02893. URL http://arxiv.org/abs/1803.02893.

Andrew Maas, Raymond Daly, Peter Pham, Dan Huang, Andrew Ng and Christopher Potts (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150. Association for Computational Linguistics, Portland, Oregon, USA. URL http://www.aclweb.org/anthology/P11-1015.

Tamar Makharoblidze (?). The Georgian verb. URL http://eprints.iliauni.edu.ge/3038/1/TheGeorgianVerb.pdf.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz and Britta Schasberger (1994). The Penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT 1994, pages 114–119. Association for Computational Linguistics, Stroudsburg, PA, USA. ISBN 1-55860-357-3. doi:10.3115/1075812.1075835. URL https://doi.org/10.3115/1075812.1075835.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi and Roberto Zamparelli (**2014**). A SICK cure for the evaluation of compositional distributional semantic models. URL http://www.lrec-conf.org/proceedings/lrec2014/pdf/363_Paper.pdf.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló and Jungmee Lee (**2013**). Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97. Association for Computational Linguistics, Sofia, Bulgaria. URL https://www.aclweb.org/anthology/P13-2017.

Yashar Mehdad, Matteo Negri and Marcello Federico (**2011**). Using bilingual parallel corpora for cross-lingual textual entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT 2011, pages 1336–1345. Association for Computational Linguistics, Stroudsburg, PA, USA. ISBN 978-1-932432-87-9. URL http://dl.acm.org/citation.cfm?id=2002472.2002638.

Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean (**2013**a). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781. URL http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781.

Tomas Mikolov, Wen-tau Yih and Geoffrey Zweig (**2013**b). Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751. URL https://www.microsoft.com/en-us/research/publication/linguistic-regularities-in-continuous-space-word-representations/.

George Miller (**1995**). WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41. URL http://www1.cs.columbia.edu/~vh/courses/LexicalSemantics/Ontologies/miller-wordnet95.pdf.

Marie-Francine Moens (**2013**). Argumentation mining: Where are we now, where do we want to be and how do we get there? volume 04-06-December-2013. ACM; New York. ISBN 9781450328302. URL https://lirias.kuleuven.be/retrieve/283641$$DMoensFIRE2014.pdf.

Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau and Chris Reed (**2007**). Automatic detection of arguments in legal texts. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law*, ICAIL 2007, pages 225–230. ACM, New York, NY, USA. ISBN 978-1-59593-680-6. doi:10.1145/1276318.1276362. URL http://doi.acm.org/10.1145/1276318.1276362.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen and Steve Young (**2016**). Counter-fitting word vectors to linguistic constraints. In *HLT-NAACL*. URL http://arxiv.org/abs/1603.00892.

Nikola Mrkšić, Ivan Vulic, Diarmuid Séaghdha, Ira Leviant, Roi Reichart, Milica Gasic, Anna Korhonen and Steve Young (**2017**). Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *CoRR*, abs/1706.00374. URL http://arxiv.org/abs/1706.00374.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Penstein Rosé and Graham Neubig (**2018**). Stress test evaluation for natural language inference. *CoRR*, abs/1806.00692. URL http://arxiv.org/abs/1806.00692.

Sascha Narr, Michael Hülfenhaus and Sahin Albayrak (**2012**). Language-independent twitter sentiment analysis. URL http://www.dai-labor.de/fileadmin/files/publications/narr-twittersentiment-KDML-LWA-2012.pdf.

Matteo Negri, Luisa Bentivogli, Yashar Mehdad, Danilo Giampiccolo and Alessandro Marchetti (**2011**). Divide and conquer: Crowdsourcing the creation of cross-lingual textual entailment corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP 2011, pages 670–679. Association for Computational Linguistics, Stroudsburg, PA, USA. ISBN 978-1-937284-11-4. URL http://dl.acm.org/citation.cfm?id=2145432.2145510.

Alvaro Ortigosa, José M. Martín and Rosa M. Carro (**2014**). Sentiment analysis in Facebook and its application to e-learning. *Computers in Human Behavior*, 31:527–541. ISSN 0747-5632. doi:10.1016/j.chb.2013.05.024. URL http://dx.doi.org/10.1016/j.chb.2013.05.024.

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. Association for Computational Linguistics, New Orleans, LA, USA. doi:10.18653/v1/N18-1049. URL https://www.aclweb.org/anthology/N18-1049.

Alexander Pak and Patrick Paroubek (2010). Twitter as a corpus for sentiment analysis and opinion mining. volume 2010, pages 1320–1326. URL http://www.mendeley.com/research/twitter-corpus-sentiment-analysis-opinion-mining-18/.

Raquel Palau and Marie-Francine Moens (2011). Argumentation mining. *Artificial Intelligence Law*, 19(1):1–22. ISSN 0924-8463. doi:10.1007/s10506-010-9104-x. URL http://dx.doi.org/10.1007/s10506-010-9104-x.

Joonsuk Park and Claire Cardie (2014). Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38. Association for Computational Linguistics, Baltimore, MD. doi:10.3115/v1/W14-2105. URL https://www.aclweb.org/anthology/W14-2105.

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. Association for Computational Linguistics, Doha, Qatar. doi:10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

Christian Perone, Roberto Silveira and Thomas Paula (2018). Evaluation of sentence embeddings in downstream and linguistic probing tasks. *CoRR*, abs/1806.06259. URL https://arxiv.org/abs/1806.06259.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee and Luke Zettlemoyer (2018). Deep contextualized word representations. URL http://arxiv.org/abs/1802.05365.

Yuanyuan Qiu, Hongzheng Li, Shen Li, Yingdi Jiang, Renfen Hu and Lijiao Yang (2018). Revisiting correlations between intrinsic and extrinsic evaluations of word embeddings. In *CCL*, volume 11221 of *Lecture Notes in Computer Science*, pages 209–221. Springer. URL http://www.cips-cl.org/static/anthology/CCL-2018/CCL-18-086.pdf.

Alec Radford, Karthik Narasimhan, Tim Salimans and Ilya Sutskever (2018). Improving language understanding by generative pre-training. URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

Vinit Ravishankar, Lilja Øvrelid and Erik Velldal (2019). Probing multilingual sentence representations with X-probe. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 156–168. Association for Computational Linguistics, Florence, Italy. URL https://www.aclweb.org/anthology/W19-4318.

Nils Reimers and Iryna Gurevych (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. URL http://arxiv.org/abs/1908.10084.

Tilmann Reuther (2009). Sprachenstreckbrief Russisch. URL http://www.schule-mehrsprachig.at/fileadmin/schule_mehrsprachig/redaktion/sprachensteckbriefe/SSB_2019/russisch.pdf.

Anna Rogers, Alexey Romanov, Anna Rumshisky, Svitlana Volkova, Mikhail Gronas and Alex Gribov (2018). RuSentiment: An enriched sentiment analysis dataset for social media in Russian. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 755–763. Association for Computational Linguistics, Santa Fe, NM, USA. URL https://www.aclweb.org/anthology/C18-1064.

Andreas Rücklé, Steffen Eger, Maxime Peyrard and Iryna Gurevych (2018). Concatenated p-mean word embeddings as universal cross-lingual sentence representations. *CoRR*, abs/1803.01400. URL http://arxiv.org/abs/1803.01400.

Piotr Rybak and Alina Wróblewska (2018). Semi-supervised neural system for tagging, parsing and lematization. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 45–54. Association for Computational Linguistics, Brussels, Belgium. doi:10.18653/v1/K18-2004. URL https://www.aclweb.org/anthology/K18-2004.

Gözde Sahin, Clara Vania, Ilia Kuznetsov and Iryna Gurevych (2019). LINSPECTOR: Multilingual probing tasks for word representations. *CoRR*, abs/1903.09442. URL http://arxiv.org/abs/1903.09442.

Manuela Sanguinetti and Christina Bosco (2014a). Converting the parallel treebank ParTUT in universal stanford dependencies. In *In proceedings of the 1rst Conference for Italian Computational Linguistics (CLiC-it 2014)*. URL http://clic2014.fileli.unipi.it/proceedings/vol1/CLICIT2014161.pdf.

Manuela Sanguinetti and Christina Bosco (2014b). PartTUT: The Turin University parallel treebank. In *Harmonization and development of resources and tools for Italian Natural Language Processing within the PARLI project*. Springer Verlag. URL https://link.springer.com/chapter/10.1007/978-3-319-14206-7_3.

Hinrich Schütze (**1993**). Word space. In *Advances in Neural Information Processing Systems 5*, pages 895–902. Morgan-Kaufmann. URL http://papers.nips.cc/paper/603-word-space.pdf.

Holger Schwenk (**2018**). Filtering and mining parallel data in a joint multilingual space. *CoRR*, abs/1805.09822. URL http://arxiv.org/abs/1805.09822.

Tatiana Shavrina and Olga Shapovalova (**2017**). To the methodology of corpus construction for machine learning: 'Taiga' syntax tree corpus and parser. In *In proceedings of the International Conference 'CORPORA 2017'*.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao and Lawrence Carin (**2018**). Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. *CoRR*, abs/1805.09843. URL https://arxiv.org/abs/1805.09843.

Xing Shi, Inkit Padhi and Kevin Knight (**2016**). Does string-based neural MT learn source syntax? In *EMNLP*, pages 1526–1534. The Association for Computational Linguistics. URL https://aclweb.org/anthology/D16-1159/.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer and Christopher Manning (**2014**). A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. URL https://nlp.stanford.edu/pubs/Gold_LREC14.pdf.

Christian Stab, Tristan Miller, Benjamin Schiller, Pranav Rai and Iryna Gurevych (**2018**). Cross-topic argument mining from heterogeneous sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3664–3674. Association for Computational Linguistics, Brussels, Belgium. doi:10.18653/v1/D18-1402. URL https://www.aclweb.org/anthology/D18-1402.

Umut Sulubacak, Memduh Gokirmak, Francis M. Tyers, Çagri Çöltekin, Joakim Nivre and Gülsen Eryigit (**2016**). Universal dependencies for Turkish. In *COLING*, pages 3444–3454. ACL. ISBN 978-4-87974-702-0. URL http://dblp.uni-trier.de/db/conf/coling/coling2016.html#SulubacakGTCNE16.

Ann Taylor, Mitchell Marcus and Beatrice Santorini (**2003**). The Penn treebank: An overview. doi:10.1007/978-94-010-0201-1_1. URL https://www.researchgate.net/publication/2873803_The_Penn_Treebank_An_overview.

Wilson Taylor (**1953**). 'Cloze Procedure': A new tool for measuring readability. volume 30, pages 415–433. doi:10.1177/107769905303000401. URL https://journals.sagepub.com/doi/abs/10.1177/107769905303000401.

Alaettin Uçan, Behzad Naderalvojoud, Ebru Sezer and Hayri Sever (**2016**). SentiWordNet for new language: Automatic translation approach. doi:10.1109/SITIS.2016.57. URL https://www.researchgate.net/publication/316241544_SentiWordNet_for_New_Language_Automatic_Translation_Approach.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser and Illia Polosukhin (**2017**). Attention is all you need. *CoRR*, abs/1706.03762. URL https://arxiv.org/abs/1706.03762.

Sara Veldhoen, Dieuwke Hupkes and Willem Zuidema (**2016**). Diagnostic classifiers: Revealing how neural networks process hierarchical structure. In *CoCo@NIPS*. URL https://www.researchgate.net/publication/311545111_Diagnostic_classifiers_revealing_how_neural_networks_process_hierarchical_structure.

Alexander von Humboldt (**1836**). *Ueber die Verschiedenheit des menschlichen Sprachbaues und ihren Einfluss auf die geistige Entwicklung des Menschengeschlechts*. Dr. der königlichen Akadademie der Wissenschaften. URL https://books.google.de/books?id=quxJAAAAcAAJ.

Ellen Voorhees and Dawn Tice (**2000**). Building a question answering test collection. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2000, pages 200–207. ACM, New York, NY, USA. ISBN 1-58113-226-3. doi:10.1145/345508.345577. URL http://doi.acm.org/10.1145/345508.345577.

Ivan Vulic, Nikola Mrksic, Roi Reichart, Diarmuid Séaghdha, Steve Young and Anna Korhonen (**2017**). Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. *CoRR*, abs/1706.00377. URL http://arxiv.org/abs/1706.00377.

John Wieting, Mohit Bansal, Kevin Gimpel and Karen Livescu (**2016**a). Charagram: Embedding words and sentences via character n-grams. *CoRR*, abs/1607.02789. URL http://arxiv.org/abs/1607.02789.

John Wieting, Mohit Bansal, Kevin Gimpel and Karen Livescu (**2016**b). Towards universal paraphrastic sentence embeddings. In *ICLR*. URL https://arxiv.org/abs/1511.08198.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu and Dan Roth (**2015**). From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358. URL http://www.cs.cmu.edu/~jwieting/wieting2015TACL.pdf.

John Wieting and Douwe Kiela (**2019**). No training required: Exploring random encoders for sentence classification. *CoRR*, abs/1901.10444. URL http://arxiv.org/abs/1901.10444.

Adina Williams, Nikita Nangia and Samuel Bowman (**2018**). A broad-coverage challenge corpus for sentence understanding through inference. *CoRR*, abs/1704.05426. URL https://arxiv.org/abs/1704.05426.

Ronald Williams and David Zipser (**1998**). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1. doi:10.1162/neco.1989.1.2.270. URL https://dl.acm.org/citation.cfm?id=1351135.

Michael Wojatzki, Eugen Ruppert, Sarah Holschneider, Torsten Zesch and Chris Biemann (**2017**). GermEval 2017: Shared task on aspect-based sentiment in social media customer feedback. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 1–12. Berlin, Germany. URL https://duepublico2.uni-due.de/servlets/MCRFileNodeServlet/duepublico_derivate_00045683/GermEval2017_Proceedings.pdf.

Ziyi Yang, Chenguang Zhu and Weizhu Chen (**2018**). Zero-training sentence embedding via orthogonal basis. *CoRR*, abs/1810.00438. URL http://arxiv.org/abs/1810.00438.

Helen Yannakoudakis, Ted Briscoe and Ben Medlock (**2011**). A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 180–189. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P11-1019/.

Amir Zeldes (**2017**). The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612. doi:http://dx.doi.org/10.1007/s10579-016-9343-x. URL https://link.springer.com/article/10.1007%2Fs10579-016-9343-x.

Xinjie Zhou, Xiaojun Wan and Jianguo Xiao (**2016**). Cross-lingual sentiment classification with bilingual document representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1412. Association for Computational Linguistics, Berlin, Germany. doi:10.18653/v1/P16-1133. URL https://www.aclweb.org/anthology/P16-1133.

Xunjie Zhu, Tingfeng Li and Gerard de Melo (**2018**). Exploring semantic properties of sentence embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers*, pages 632–637. Association for Computational Linguistics, Melbourne, Australia. URL https://www.aclweb.org/anthology/P18-2100.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba and Sanja Fidler (**2015**). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, abs/1506.06724. URL http://arxiv.org/abs/1506.06724.

Webpages

https://1000mostcommonwords.com/ (retrieved: September 08, 2019)

https://achyutjoshi.github.io/aspect_extraction/aspectextraction (retrieved: October 16, 2019)

https://blackboxnlp.github.io/ (retrieved: October 04, 2019)

https://cogcomp.seas.upenn.edu/page/resource_view/49 (retrieved: September 08, 2019)

https://www.colanguage.com/turkish-adjectives (retrieved: October 02, 2019)

https://corpora.uni-hamburg.de/hzsk/de/islandora/object/treebank:hdt (retrieved: September 07, 2019)

https://deacademic.com/dic.nsf/dewiki/510990 (retrieved: October 02, 2019)

https://dumps.wikimedia.org/ (retrieved: September 20, 2019) https://emojipedia.org/ (retrieved: September 18, 2019)

http://gnc.gov.ge/gnc/page (retrieved: September 07, 2019)

https://www.grammatiken.de/tuerkische-grammatik/ (retrieved: October 11, 2019)

http://ftp.gwdg.de/pub/ctan/fonts/georgian/mxedruli/mxeddoc.pdf (retrieved: September 06, 2019)

https://www.ims.uni-stuttgart.de/ (retrieved: September 07, 2019)

https://www.informatik.tu-darmstadt.de/ (retrieved: September 08, 2019)

https://issuu.com/artemivantsov/docs/150_georgian_verbs (retrieved: September 07, 2019)

http://jalammar.github.io/illustrated-bert/ (retrieved: September 04, 2019)

https://www.kaggle.com/crowdflower/twitter-airline-sentiment (retrieved: September 08, 2019)

https://medium.com/ (retrieved: October 07, 2019)

https://www.mturk.com/ (retrieved: September 08, 2019)

https://www.russlandjournal.de/ (retrieved: September 06, 2019)

http://www.schule-mehrsprachig.at/ (retrieved: September 06, 2019)

https://nlp.stanford.edu/projects/snli/ (retrieved: September 04, 2019)

https://towardsdatascience.com/ (retrieved: October 07, 2019)

http://ge.translit.cc/ (retrieved: September 07, 2019)

https://public.ukp.informatik.tu-darmstadt.de/ (retrieved: September 04, 2019)

https://uni-tuebingen.de/de/134290 (retrieved: September 09, 2019)

https://universaldependencies.org/ (retrieved: September 07, 2019)

https://de.wikipedia.org/wiki/Georgische_Sprache (retrieved: September 06, 2019)

https://en.wikipedia.org/wiki/Turkish_language (retrieved: September 06, 2019)

http://www.wikiwand.com/fr/Arbre_syntaxique (retrieved: September 05, 2019)

GitHub Repositories

https://github.com/attardi/wikiextractor (retrieved: September 20, 2019)

https://github.com/epfml/sent2vec (retrieved: September 04, 2019)

https://github.com/facebookresearch/InferSent (retrieved: September 04, 2019)

https://github.com/facebookresearch/LASER (retrieved: September 04, 2019)

https://github.com/facebookresearch/SentEval (retrieved: September 03, 2019)

https://github.com/google-research/bert/blob/master/multilingual.md (retrieved: September 04, 2019)

https://github.com/lajanugen/S2V (retrieved: September 04, 2019)

https://github.com/tmikolov/word2vec (retrieved: September 20, 2019)

https://github.com/UKPLab/arxiv2018-xling-sentence-embeddings (retrieved: September 09, 2019)

Videos

https://www.youtube.com/watch?v=ERibwqs9p38 (retrieved: September 03, 2019)

Images

https://blog.f1000.com/wp-content/uploads/2017/06/black_box_blog.png (retrieved: September 04, 2019)