ARTIFICIAL NEURAL NETWORK METHODS APPLIED TO SENTIMENT ANALYSIS

Inaugural-Dissertation
zur Erlangung des Doktorgrades der Philosophie
an der Ludwig-Maximilians-Universität München

vorgelegt von
Sebastian Ebert
aus Ilmenau

München 2017

Referent: Prof. Dr. Hinrich Schütze
Korreferent: Dr. Helmut Schmid

Tag der mündlichen Prüfung: 07.02.2017

ABSTRACT

Sentiment Analysis (SA) is the study of opinions and emotions that are conveyed by text. This field of study has commercial applications for example in market research (e.g., "What do customers like and dislike about a product?") and consumer behavior (e.g., "Which book will a customer buy next when he wrote a positive review about book X?"). A private person can benefit from SA by automatic movie or restaurant recommendations, or from applications on the computer or smart phone that adapt to the user's current mood.

In this thesis we will put forward research on artificial Neural Network (NN) methods applied to SA. Many challenges arise, such as sarcasm, domain dependency, and data scarcity, that need to be addressed by a successful system.

In the first part of this thesis we perform linguistic analysis of a word ("hard") under the light of SA. We show that sentiment-specific word sense disambiguation is necessary to distinguish fine nuances of polarity. Commonly available resources are not sufficient for this.

The introduced Contextually Enhanced Sentiment Lexicon (CESL) is used to label occurrences of "hard" in a real dataset with its sense. That allows us to train a Support Vector Machine (SVM) with deep learning features that predicts the polarity of a single occurrence of the word, just given its context words. We show that the features we propose improve the result compared to existing standard features. Since the labeling effort is not neglectible, we propose a clustering approach that reduces the manual effort to a minimum.

The deep learning features that help predicting fine-grained, context-dependent polarity are computed by a Neural Network Language Model (NNLM), namely a variant of the Log-Bilinear Language model (LBL). By improving this model the performance of polarity classification might as well improve. Thus, we propose a non-linear version of the LBL and the vectorized Log-Bilinear Language model (vLBL), because non-linear models are generally considered more powerful. In a parameter study on a language modeling task, we show that the non-linear versions indeed perform better than their linear counterparts. However, the difference is small, except for settings where the model has only few parameters, which might be the case when little training data is available and the model therefore needs to be smaller in order to avoid overfitting.

An alternative approach to fine-grained polarity classification as used above is to train classifiers that will do the distinction automatically. Due to the complexity of the task, the challenges of SA in general, and certain domain-specific issues (e.g., when using Twitter text) existing systems have much room to improve. Often statistical classifiers are used with simple Bag-of-Words (BOW) features or count features that stem from sentiment lexicons. We introduce a linguistically-informed Convolutional Neural Network (lingCNN) that builds upon the fact that there has been much research on language in general and sentiment lexicons in particular. lingCNN makes use of two types of linguistic features: word-based and sentence-based. Word-based features comprise features derived from sentiment lexicons, such as polarity or valence and general knowledge about language, such as a negation-based feature. Sentence-based features are also based on lexicon counts and valences. The combination of both types of features is superior to the original model without these features. Especially, when little training data is available (that can be the case for different languages that are underresourced), lingCNN proves to be significantly better (up to 12 macro-F_1 points).

Although, linguistic features in terms of sentiment lexicons are beneficial, their usage gives rise to a new set of problems. Most lexicons consist of infinitive forms of words only. Especially, lexicons for low-resource languages. However, the text that needs to be classified is unnormalized. Hence, we want to answer the question if morphological information is necessary for SA or if a system that neglects all this information and therefore can make better use of lexicons actually has an advantage. Our approach is to first stem or lemmatize a dataset and then perform polarity classification on it. On Czech and English datasets we show that better results can be achieved with normalization. As a positive side effect, we can compute better word embeddings by first normalizing the training corpus. This works especially well for languages that have rich morphology. We show on word similarity datasets for English, German, and Spanish that our embeddings improve performance. On a new WordNet-based evaluation we confirm these results on five different languages (Czech, English, German, Hungarian, and Spanish). The benefit of this new evaluation is further that it can be used for many other languages, as the only resource that is required is a WordNet.

In the last part of the thesis, we use a recently introduced method to create an ultradense sentiment space out of generic word embeddings. This method allows us to compress 400 dimensional word embeddings down to 40 or even just 4 dimensions and still get similar results on a polarity classification task. While the training speed increases by a factor of 44, the difference in classification performance is not significant.

ABSTRAKT

Sentiment Analyse (SA) ist das Untersuchen von Meinungen und Emotionen die durch Text übermittelt werden. Dieses Forschungsgebiet findet kommerzielle Anwendungen in Marktforschung (z.B.: „Was mögen Kunden an einem Produkt (nicht)?") und Konsumentenverhalten (z.B.: „Welches Buch wird ein Kunde als nächstes kaufen, nachdem er eine positive Rezension über Buch X geschrieben hat?"). Aber auch als Privatperson kann man von Forschung in SA profitieren. Beispiele hierfür sind automatisch erstellte Film- oder Restaurantempfehlungen oder Anwendungen auf Computer oder Smartphone die sich der aktuellen Stimmungslage des Benutzers anpassen.

In dieser Arbeit werden wir Forschung auf dem Gebiet der Neuronalen Netze (NN) angewendet auf SA vorantreiben. Dabei ergeben sich viele Herausforderungen, wie Sarkasmus, Domänenabhängigkeit und Datenarmut, die ein erfolgreiches System angehen muss.

Im ersten Teil der Arbeit führen wir eine linguistische Analyse des englischen Wortes „hard" in Hinblick auf SA durch. Wir zeigen, dass sentiment-spezifische Wortbedeutungsdisambiguierung notwendig ist, um feine Nuancen von Polarität (positive vs. negative Stimmung) unterscheiden zu können. Häufig verwendete, frei verfügbare Resourcen sind dafür nicht ausreichend. Daher stellen wir CESL (Contextually Enhanced Sentiment Lexicon), ein sentiment-spezifisches Bedeutungslexicon vor, welches verwendet wird, um Vorkommen von „hard" in einem realen Datensatz mit seinen Bedeutungen zu versehen. Das Lexicon erlaubt es eine Support Vector Machine (SVM) mit Features aus dem Deep Learning zu trainieren, die in der Lage ist, die Polarität eines Vorkommens nur anhand seiner Kontextwörter vorherzusagen. Wir zeigen, dass die vorgestellten Features die Ergebnisse der SVM verglichen mit Standard-Features verbessern. Da der Aufwand für das Erstellen von markierten Trainingsdaten nicht zu unterschätzen ist, stellen wir einen Clustering-Ansatz vor, der den manuellen Markierungsaufwand auf ein Minimum reduziert.

Die Deep Learning Features, die die Vorhersage von feingranularer, kontextabhängiger Polarität verbessern, werden mittels eines neuronalen Sprachmodells, genauer eines Log-Bilinear Language model (LBL)s, berechnet. Wenn man dieses Modell verbessert, wird vermutlich auch das Ergebnis der Polaritätsklassifikation verbessert. Daher führen wir nichtlineare Versionen des LBL und vectorized Log-Bilinear Language model (vLBL) ein, weil nichtlineare Modelle generell als mächtiger angesehen werden. In einer Parameterstudie zur Sprachmodellierung zeigen wir, dass nichtlineare Modelle tatsächlich besser abschneiden, als ihre linearen Gegenstücke. Allerdings ist der Unterschied gering, es sei denn die Modelle können nur auf wenige Parameter zurückgreifen. So etwas kommt zum Beispiel vor, wenn nur wenige Trainingsdaten verfügbar sind und das Modell deshalb kleiner sein muss, um Überanpassung zu verhindern.

Ein alternativer Ansatz zur feingranularen Polaritätsklassifikation wie oben verwendet, ist es, einen Klassifikator zu trainieren, der die Unterscheidung automatisch vornimmt. Durch die Komplexität der Aufgabe, der Herausforderungen von SA im Allgemeinen und speziellen domänenspezifischen Problemen (z.B.: wenn Twitter-Daten verwendet werden) haben existierende Systeme noch immer großes Optimierungspotential. Oftmals verwenden statistische Klassifikatoren einfache Bag-of-Words (BOW)-Features. Alternativ kommen Zähl-Features zum Einsatz, die auf Sentiment-Lexika aufsetzen. Wir stellen linguistically-informed Convolutional Neural Network (lingCNN) vor, dass auf dem Fakt beruht, dass bereits viel Forschung in Sprachen und Sentiment-Lexika geflossen ist. lingCNN macht von zwei linguistischen Feature-Typen Gebrauch: wortbasierte und satzbasierte. Wortbasierte Features umfassen Features die von Sentiment-Lexika, wie Polarität oder Valenz (die Stärke der Polarität) und generellem Wissen über Sprache, z.B.: Verneinung, herrühren. Satzbasierte Features basieren ebenfalls auf Zähl-Features von Lexika und auf Valenzen. Die Kombination beider Feature-Typen ist dem Originalmodell ohne linguistische Features überlegen. Besonders wenn wenige Trainingsdatensätze vorhanden sind (das kann der Fall für Sprachen sein, die weniger erforscht sind als englisch). lingCNN schneidet signifikant besser ab (bis zu 12 macro-F_1 Punkte).

Obwohl linguistische Features basierend auf Sentiment-Lexika vorteilhaft sind, führt deren Verwendung zu neuen Problemen. Der Großteil der Lexika enthält nur Infinitivformen der Wörter. Dies gilt insbesondere für Sprachen mit wenigen Resourcen. Das ist eine Herausforderung, weil der Text der klassifiziert werden soll in der Regel nicht normalisiert ist. Daher wollen wir die Frage beantworten, ob morphologische Information für SA überhaupt notwendig ist oder ob ein System, dass jegliche morphologische Information ignoriert und dadurch bessere Verwendung der Lexika erzielt, einen Vorteil genießt. Unser Ansatz besteht aus Stemming und Lemmatisierung des Datensatzes, bevor dann die Polaritätsklassifikation durchgeführt wird. Auf englischen und tschechischen Daten zeigen wir, dass durch Normalisierung bessere Ergebnisse erzielt werden. Als positiven Nebeneffekt kann man bessere Wortrepresentationen (engl. word embeddings) berechnen, indem das Trainingskorpus zuerst normalisiert wird. Das funktioniert besonders gut für morphologisch reiche Sprachen. Wir zeigen auf Datensätzen zur Wortähnlichkeit für deutsch, englisch und spanisch, dass unsere Wortrepresentationen die Ergebnisse verbessern. In einer neuen WordNet-basierten Evaluation bestätigen wir diese Ergebnisse für fünf verschiedene Sprachen (deutsch, englisch, spanisch, tschechisch und ungarisch). Der Vorteil dieser Evaluation ist weiterhin, dass sie für viele Sprachen angewendet werden kann, weil sie lediglich ein WordNet als Resource benötigt.

Im letzten Teil der Arbeit verwenden wir eine kürzlich vorgestellte Methode zur Erstellen eines ultradichten Sentiment-Raumes aus generischen Wortrepresentationen. Diese Methode erlaubt es uns 400 dimensionale Wortrepresentationen auf 40 oder sogar nur 4 Dimensionen zu komprimieren und weiterhin die gleichen Resultate in Polaritätsklassifikation zu erhalten. Während die Trainingsgeschwindigkeit um einen Faktor von 44 verbessert wird, sind die Unterschiede in der Polaritätsklassifikation nicht signifikant.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Sentiment Analysis (SA) deals with the recognition of a person's opinion towards a specific topic or product, or a property thereof (Pang and Lee, 2008). Whenever an automated analysis of a person's opinion or feeling towards something is requested, SA techniques come into play. More specifically, politicians might want to know what their voters think about a specific bill they advertise or a company might want to get an idea about product users in order to learn about problems and maybe possible improvements. But also an individual user can benefit from an automated system, for instance when there is a need for recommendations for products, vacations, etc. A movie fan can get recommendations for a "sad movie with happy end". Besides that there are other useful applications of SA. For example, it enables the summarization of opinions regarding a topic, may it be textual or visual. Further, a well working system can be used to correct for false labeled data in cases where a user's star rating does not align with the written text. An intriguing application are question answering systems that can be enabled to react according to a user's mood and might soothe them when being sad.

This thesis focuses on text in contrast to sentiment in speech or videos. Furthermore, we cover only *polarity classification*, the classification of text into coarse categories, mostly into *positive*, *negative*, and *neutral*. A different approach would be to classify into more fine-grained categories, such as a 5-star rating with strongly negative, slightly negative, neutral, slightly positive, and strongly positive. Extending this idea leads to analyzing a text's *valence*, i.e., the magnitude of positivity or negativity on a continuous range of values. This however is not the focus of this work.

Furthermore, we deal with the polarity of entire pieces of text, not depending on specific properties or *aspects* of products. For example, we want to know if a given text is overall positive instead of which aspect of a product is positive.

Finally, emotion research is also not in the focus of this work. For instance, we do not want to classify a reviewer's mood into basic emotions such as the 8 basic emotions of Plutchik, (1980).

1.1 CHALLENGES

Major challenges research in SA faces are:

DOMAIN DEPENDENCY Polarity is domain dependent. Consider the sentence: "Go, read the book!" When talking about a book, this statement is positive. However, when talking about a movie that is based on a book, this statement is negative.

MULTIPLE ASPECTS One review can comprise multiple aspects about an item. For instance, a reviewer might write about actors of a movie that he did not like, but still might like the movie overall.

MULTIPLE WORD SENSES Many words carry more than one sense. For polarity classification this is challenging, because in one occurrence a word can be polar, whereas in another context the same word is neutral.

DATA SPARSITY Labeled sentiment datasets are scarce. Especially, when aspect-level labels are required much manual effort is necessary to create a dataset that can be used in statistical approaches. But also coarse-grained sentence-level labels are scarce, especially in languages other than English.

LINGUISTIC PROBLEMS Such problems include spelling errors (e.g., "wierd" instead of "weird"), colloquial speech (e.g., "swell" instead of "great"), and frequently used emoticons (e.g., ":)" meaning something is positive). In contrast to these more word-based problems there are problems arising out of composition. Negation often (but not always) changes the entire polarity of a sentence. Valence shifters (Polanyi and Zaenen, 2004; 2006) change the valence of polarity. Some problems only arise in certain domains. In Twitter for example, user's are forced to heavily abbreviate words due to the 140 character limit per message. Texts such as "whr go sux? life is sooo beautiful !" are very common.

1.2 EXISTING APPROACHES

1.2.1 *Lexicon Creation*

Much research in SA has gone into lexicon creation. The underlying idea is that words have a *prior polarity*, which is the polarity one would assign a word when nothing about the word's context is known. Manual approaches for creating lexicons comprise for example the General Inquirer lexicon (Stone et al., 1966), ANEW (Bradley and Lang, 1999), and the lexicons created by Taboada et al., (2011). Semi-automatically created lexicons, usually based on a list of seed words were used among others by Mohammad et al., (2013), Turney, (2002), and Wilson et al., (2005).

Sentiment lexicons have been used in many approaches for SA, ranging from Bayesian approaches (Maas et al., 2011) and machine learning approaches (e.g., Tang, Wei, Qin, Zhou, et al., (2014) and Tang, Wei, Yang, et al., (2014)), to systems that explicitly use linguistic features (Gamon, 2004) and other types of polarity features such as valence shifters (Taboada et al., 2011).

1.2.2 *Statistical Classification Methods*

On the other hand there are attempts to learn statistical classifiers without any additional linguistic knowledge such as sentiment lexicons. One of the first attempts of automated polarity classification tried to classify movie reviews into positive, negative, and neutral (Pang et al., 2002). The labels for the created dataset were automatically extracted from user-provided star ratings and thus may be noisy. The authors used Bag-of-Words (BOW) features with different weighting schemes (binary and frequency) and a position feature plus a simple negation detection in Naive Bayes, Maximum Entropy, and Support Vector Machine (SVM) models. Surprisingly, a simple binary unigram SVM yielded the best performance. S. I. Wang and Manning, (2012) later found that bigram features consistently improve the performance. They showed improvements for all three of their tested classifiers (Naive Bayes, SVM, and Naive Bayes SVM).

This straight-forward classification approach was followed by a two step approach where first a classifier determines if a sentence in the review is *subjective* or *objective* and then in a second step another classifier classifies the polarity of only the subjective sentences (Pang and Lee, 2004; Riloff, Wiebe, Collins, et al., 2003).

Scheible and Schütze, (2013) later argued that classifying only subjective and objective sentences is not appropriate for detailed SA, because there are subjective sentences that do not convey sentiment and there are objective sentences that do convey polarity.

Recently, most work aims for implicit and automatic learning of subjectivity or relevance by having a single classifier for the text's polarity without explicit modeling of both (e.g., Hagen et al., (2015)). More recent approaches to SA include neural network methods such as Recursive Neural Networks (Socher et al., 2013) and Convolutional Neural Networks (CNNs) (Kim, 2014; Severyn and Moschitti, 2015; Yin and Schütze, 2015).

In (Meng et al., 2015) an ensemble of ngram Language Models (LMs), Recurrent Neural Network (RNN) LM (Mikolov et al., 2010), sentence vectors (Le and Mikolov, 2014), and a Naive Bayes SVM was shown to reach state-of-the-art performance on a large sentiment treebank (Socher et al., 2013). In their study, the ngram LMs had the smallest effect on the results. All other classifiers contributed to the final result in the linear interpolation.

1.2.3 *Word Representation Learning*

The following approaches learn special word representations to support SA:

Maas and Ng, (2010) introduce a probabilistic model that learns semantic, distributed word representations. Instead of focusing on syntax during word representation induction the model takes document term relations into account, i.e., it uses the correlation of words within documents inside the training objective, leading to more semantic word representations. These capture the "empirical distribution of words in a document". The presented approach is similar to LDA, but learns word representations instead of topic distributions. The power of the word embeddings is evaluated on document level sentiment classification and sentence level subjectivity detection.

Maas et al., (2011) follow up on the idea of semantic word representations and propose a model that learns word representations that capture semantic similarities and word sentiment at the same time in a multi-task learning setting. This is done by having a training objective that combines a semantic training objective taken from Maas and Ng, (2010) and a sentiment-based training objective, which is a classification score on movie review data.

Labutov and Lipson, (2013) introduce a method to use existing source representations, i.e., representations that have been computed by external parties, to improve supervised tasks. The idea is to have a supervised training objective for the task at hand that learns target representations optimized for this task and an objective that makes use of the source representations to guide the learning of the target representations by computing a norm of the representations' differences. The usage of different source representations (Collobert et al., 2011; Huang et al., 2012; Mnih and Hinton, 2008) and different training set sizes shows superior performance on a sentiment classification task.

Tang, Wei, Yang, et al., (2014) present 3 different neural networks for learning sentiment specific n-gram representations. The models are based on Collobert et al., (2011)'s model and incorporate sentiment labels into the loss function. All sentiment labels are extracted automatically from Twitter tweets by searching for predefined emoticons in the texts and labelling them accordingly, e.g., ":-)" is positive. Their best model combines the hinge loss of Collobert et al., (2011)'s model and the sentiment hinge loss. Evaluation on the SemEval 2013 test set shows superior performance compared to all baselines, including the SemEval2013 winner. Further evaluations show that using higher-order n-gram embeddings helps in classification.

1.3 OUTLINE AND CONTRIBUTIONS

Our contributions align with the chapters in this thesis. The chapters and main contributions per chapter are:

CHAPTER 2 gives an overview over basic concepts and methods that are used throughout this thesis. It describes language modeling as a Natural Language Processing (NLP) task and is used in Chapter 3 and Chapter 4. It further gives an overview over CNNs that are later extended in Chapter 5.

CHAPTER 3 introduces a system for fine-grained polarity classification. We argue that fine-grained polarity depends on the sense of a word, which depends on the context. Therefore, we analyze the senses of a word in the light of sentiment and show that sentiment specific senses are different than senses in terms of linguistic meaning. Building upon this sense inventory we propose a method for context-dependent polarity classification, based on either an LM or a clustering method.

CHAPTER 4 builds upon the idea used in Chapter 3 to use an LM for polarity classification. In order to improve language modeling we introduce non-linear extensions to the linear Log-Bilinear Language model (LBL) that are easily applicable to other linear models. We reach the biggest gains when a model has only a small number of parameters, leading to benefits for settings with little training data.

CHAPTER 5 describes the incorporation of linguistic knowledge into a CNN architecture. We present two orthogonal approaches for feature integration, a word-based and a sentence-based approach. On sentence level we show that this additional information is very beneficial for polarity classification. We show especially strong improvements when training data is scarce.

CHAPTER 6 analyzes the impact of morphological normalization on polarity classification. We compare the standard form-based approach with stem- and lemma-based methods for up to five different languages. We present strong improvements especially for Morphologically Rich Languages (MRLs), because normalization for them is more beneficial compared to languages with simple morphology, such as English. We further show that the presented methods successfully address sparsity problems and are beneficial when little training data is available.

CHAPTER 7 describes a method that converts generic word embeddings in a way that puts focused information, such as valence and concreteness, into specific dimensions. This allows us to create one-dimensional representations of words, i.e., a lexicon, for each of these properties. The created lexicons have high-quality and a large coverage. Moreover, as we show in this chapter, reducing the number of dimensions from 400 down to 40 or even 4, does not reduce the polarity classification performance much.

CHAPTER 8 concludes this thesis.

All main chapters contain their own specific introduction, detailed contribution list, related work section, and conclusion.

2 FOUNDATIONS

In this Chapter we introduce basic concepts that are required by the topics in this thesis. We start with language modeling in Section 2.1, where we describe the basic concept of Language Models, with the most prominent example, n-gram models. We explain how they are trained and evaluated and introduce a more recent Neural Network Language Model. In Section 2.2 we describe a special type of Neural Network, namely the Convolutional Neural Network, which has originally been developed for vision, but has appealing properties for natural language as well.

2.1 LANGUAGE MODELING

Language modeling is a fundamental task for many Natural Language Processing (NLP) applications such as Optical Character Recognition (OCR), Automatic Speech Recognition (ASR), statistical Machine Translation (MT), or spelling correction. Generally speaking a Language Model (LM) assigns a probability to a sequence of m words $S = w_1 w_2 \dots w_m = w_1^m$:

$$P(S) = P\left(w_1^m\right) = \prod_{i=1}^{m} P\left(w_i | w_1^{i-1}\right) \tag{2.1.1}$$

For traditional non-Neural Network (NN) LMs it is infeasible to compute $P\left(w_i | w_1^{i-1}\right)$ for large i, because there is not enough data to estimate the probability. Therefore, the Markov assumption is used in order to restrict the context to the previous $n-1$ words:

$$P\left(w_i | w_1^{i-1}\right) \approx P\left(w_i | w_{i-n+1}^{i-1}\right) = P\left(w_i | h_i\right) \tag{2.1.2}$$

Such a model is called n-gram model, where n is called the *order* of the model, and h_i is called the *history* of w_i.

2.1.1 *Training an Ngram Model*

Ngram models are learnt by Maximum Likelihood, i.e., they maximize the likelihood of a training set T given the model parameters θ:

$$\mathcal{L}(T) = P_\theta\left(T\right) = \prod_{i=1}^{M} P_\theta\left(w_i | h_i\right) \tag{2.1.3}$$

where M is the number of words in T. The Maximum Likelihood Estimate (MLE) maximizing $\mathcal{L}$ can be computed as follows:

$$P_{\mathrm{MLE}}\left(w|h\right) = \frac{c(hw)}{c(h)} \tag{2.1.4}$$

with $c(hw)$ being the frequency of the ngram hw in the training set T.

Although very simple, the MLE comes at its price. Any previously unseen ngram receives zero probability, which is clearly an underestimate. The larger n the more severe this problem becomes, because no text contains all possible ngrams (data sparsity). *Smoothing* techniques address this issue by redistributing probability mass from frequent ngrams to infrequent or unseen ngrams.

2.1.2 *Smoothing*

Chen and Goodman, (1999) give an extensive overview over many different smoothing techniques, such as additive smoothing (Laplace, 1825), Good-Turing (Good, 1953), Jelinek-Mercer (Jelinek and Mercer, 1980), Katz smoothing (Katz, 1987) etc. We focus on *modified Kneser-Ney (KN)* (Chen and Goodman, 1999), an extension to KN smoothing (Kneser and Ney, 1995) and a smoothing technique that has proven to be very powerful. This is the technique we use in later language modeling experiments.

In contrast to other smoothing techniques KN estimates lower-order ngram probabilities not on their counts but on their usage in higher order ngrams. For instance unigram probabilities are based on the number of word types they can follow, i.e., their usage in bigrams. More specifically, a unigram's probability is computed as:

$$P_{\mathrm{MKN}}\left(w_i\right) = \frac{N_{1+}\left(\bullet w_i\right)}{N_{1+}\left(\bullet\bullet\right)} \tag{2.1.5}$$

where

$$N_{1+}\left(\bullet w_i\right) = \left|\left\{w_{i-1} : c\left(w_{i-1}w_i\right) \geq 1\right\}\right| \tag{2.1.6}$$

with $c(x)$ again being the frequency of x in the training set. Hence, $N_{1+}\left(\bullet w_i\right)$ is the number of different words that precede w_i in the training set.

$$\begin{aligned} N_{1+}\left(\bullet\bullet\right) &= \sum_{w_{i-1}} N_{1+}\left(w_{i-1}\bullet\right) \\ &= \left|\left\{\left(w_{i-1}w_i\right) : c\left(w_{i-1}w_i\right) \geq 1\right\}\right| \\ &= \sum_{w_i} N_{1+}\left(\bullet w_i\right) \end{aligned} \tag{2.1.7}$$

is the number of bigram types that occur at least once.

The ngram probability of the highest order ngram depends on absolute counts:

$$P_{\text{MKN}}\left(w_i|h_i\right) = \frac{\max\left\{c\left(h_i w_i\right) - D\left(c\left(h_i w_i\right)\right), 0\right\}}{\sum_{w_i} c\left(h_i w_i\right)} + \gamma_{\text{highest}}\left(h_i\right) P_{\text{MKN}}\left(w_i|w_{i-n+2}^{i-1}\right) \tag{2.1.8}$$

This equation shows two modifications of Chen and Goodman, (1999). First, they use *interpolation* instead of *back-off* as the original implementation. This means that they use "lower-order distribution for all words, not just for words that have zero counts in the higher- order distribution" (Chen and Goodman, 1999). Second, they introduce several discount values depending on the count of a particular ngram:

$$D\left(c\right) = \begin{cases} 0 & \text{if } c = 0 \\ D_1 & \text{if } c = 1 \\ D_2 & \text{if } c = 2 \\ D_{3+} & \text{if } c \geq 3 \end{cases} \tag{2.1.9}$$

The discount parameters are computed as:

$$\begin{aligned} D_1 &= 1 - 2Y\frac{n_2}{n_1} \\ D_2 &= 2 - 3Y\frac{n_3}{n_2} \\ D_{3+} &= 3 - 4Y\frac{n_4}{n_3} \end{aligned} \tag{2.1.10}$$

where n_* are the total number of n-grams that occur exactly $*$ times and

$$Y = \frac{n_1}{n_1 + 2n_2} \tag{2.1.11}$$

The third and last modification that is introduced is that the discount parameters in Equation 2.1.10 are estimated on held-out data instead of the training set.

The normalization factor or interpolation factor that is responsible for making it sum up to 1 is defined as:

$$\gamma_{\text{highest}}\left(h_i\right) = \frac{D_1 N_1\left(h_i \bullet\right) + D_2 N_2\left(h_i \bullet\right) + D_{3+} N_{3+}\left(h_i \bullet\right)}{\sum_{w_i} c\left(h_i w_i\right)} \tag{2.1.12}$$

$$N_1\left(h_i \bullet\right) = \left|\left\{w_i : c\left(h_i w_i\right) = 1\right\}\right| \tag{2.1.13}$$

is the number of word types that follow the history exactly once. $N_2\left(h_i \bullet\right)$ and $N_{3+}\left(h_i \bullet\right)$ are defined accordingly.

The probability estimation for lower order ngrams, i.e., of order between the highest and unigram, is computed as:

$$\begin{aligned} P_{\text{MKN}}\left(w_i|h_i\right) = & \frac{\max\left\{N_{1+}\left(\bullet h_i w_i\right) - D\left(c\left(h_i w_i\right)\right), 0\right\}}{\sum_{w_i} N_{1+}\left(\bullet h_i w_i\right)} \\ & + \gamma_{\text{lower}}\left(h_i\right) P_{\text{MKN}}\left(w_i|w_{i-n+2}^{i-1}\right) \end{aligned} \quad (2.1.14)$$

where

$$\gamma_{\text{lower}}\left(h_i\right) = \frac{D_1 N_1\left(h_i \bullet\right) + D_2 N_2\left(h_i \bullet\right) + D_{3+} N_{3+}\left(h_i \bullet\right)}{\sum_{w_i} N_{1+}\left(\bullet h_i w_i\right)} \quad (2.1.15)$$

Thus, the estimated probability of a lower order ngram depends on discounts instead of absolute counts as seen for the highest order ngram.

2.1.3 *Evaluating a Language Model*

Evaluating LMs in an end to end system, such as MT or ASR is often complicated and computationally expensive. Therefore, they are often evaluated intrinsically using either cross entropy or perplexity. Cross entropy "is the average number of bits that would be required to encode the test data using an optimal coder" (Goodman, 2001):

$$\begin{aligned} H\left(D, \theta\right) &= -\sum_{hw \in V^n} P\left(hw\right) \log_2 P_\theta\left(w|h\right) \\ &= -\sum_{hw \in V^n} \frac{c(hw)}{M} \log_2 P_\theta\left(w|h\right) \\ &= -\sum_{i}^{M} \frac{1}{M} \log_2 P_\theta\left(w_i|h_i\right) \end{aligned} \quad (2.1.16)$$

where D is some dataset previously unseen and V is the vocabulary of the training set T.

Perplexity is then defined as:

$$\begin{aligned} PPL\left(D, \theta\right) &= 2^{H(D,\theta)} \\ &= 2^{-\sum_i^M \frac{1}{M} \log_2 P_\theta(w_i|h_i)} \\ &= 2^{-\frac{1}{M}\sum_i^M \log_2 P_\theta(w_i|h_i)} \\ &= \sqrt[M]{\frac{1}{\prod_i^M P_\theta\left(w_i|h_i\right)}} \end{aligned} \quad (2.1.17)$$

Minimizing perplexity corresponds to having a model that is closer to the real data distribution of D.

2.1.4 *Log-Bilinear Language Model*

As we have seen before, the larger the ngram size n the sparser the data becomes and hence the more ngram models suffer from data sparsity. There are extensions that address this problem, such as class-based LM (Brown et al., 1992) and skip-n-gram models (Pickhardt et al., 2014), that both try to cluster together ngrams (or parts thereof). A very different approach is to use NNs. First work in this direction was done by Bengio et al., (2000) and Bengio et al., (2003) who trained a feed-forward NN. Their model embeds all words into a low-dimensional real-valued space. Therefore, the model is also called Continuous Space Language Model (CSLM). The advantage of this model is that the word vectors are learned automatically in a way that words that occur in similar context get similar vectors. For instance the vector of *Saturday* will be similar to the vector of *Sunday*.

The combination of words in the continuous space as performed by the network works like implicit smoothing and can lead to non-zero probability no matter if an ngram has been seen before or not.

Low-dimensional word representations, also called *embeddings*, are created for all words in the vocabulary and are updated during the model training. This leads to similar words having similar word embeddings, which is a property that can be used in all kinds of tasks, such as word similarity judgements.

Despite its power and intriguing properties, Bengio et al., (2000)'s CSLM is computationally very expensive and needs lots of training data, because of the large number of model parameters. One way of reducing training time is by using a hierarchical version of the training algorithm as presented by Morin and Bengio, (2005). Instead of directly predicting a probability distribution over all words in the vocabulary, it predicts a bit vector representation as output. This representation encodes a path in a binary tree from the root to the leaf, where the leaf corresponds to the searched word. The advantage of this method is that there will be "gradient propagation only for the nodes on the path from the root to the leaf", which saves much computation time. This method delivers significant training speed-up, but yields lower performance than the original model.

An alternative model is the Log-Bilinear Language model (LBL) (Mnih and Hinton, 2007), another CSLM. It is a linear model, that has fewer parameters and therefore is easier to train, but reaches better performance than the CSLM from Bengio et al., (2000).

Architecture

The LBL embeds words into two distinct spaces, one input space (R) and one target space (Q), depending on whether the word occurs in the history or as a target word. We denote the input embedding of word w as $\boldsymbol{r}_w \in \mathbb{R}^d$ and the target embedding as $\boldsymbol{q}_w \in \mathbb{R}^d$, where d is the embeddings size.

For a given ngram history $h = w_1^{n-1}$ the model predicts a target embedding $\hat{\boldsymbol{q}}$ by linearly combining the context word embeddings with position dependent weights:

$$\hat{\boldsymbol{q}}_{LBL}(h) = \sum_{i=1}^{n-1} C_i \boldsymbol{r}_{w_i} \tag{2.1.18}$$

where $C_i \in \mathbb{R}^{d \times d}$ is a weight matrix associated with position i in the history. When concatenating the weight matrices ($C = C_1 \dots C_{n-1}$) and word embeddings ($r = \boldsymbol{r}_{w_1} \dots \boldsymbol{r}_{w_{n-1}}$) we see that the LBL corresponds to a NN with one hidden layer with a linear activation function:

$$\hat{\boldsymbol{q}}_{LBL}(h) = Cr \tag{2.1.19}$$

To measure the quality of the predicted target embedding $\hat{\boldsymbol{q}}_{LBL}$ and the real target embedding $\boldsymbol{q}_w$, the model computes a similarity using the dot product:

$$s_\theta(w, h) = \hat{\boldsymbol{q}}(h)^T \boldsymbol{q}_w + b_w \tag{2.1.20}$$

where b_w is a bias term and $\theta = \{R, Q, C, \boldsymbol{b}\}$ are the model's parameters. The final probability of a word given its context is calculated using the softmax function:

$$P_\theta^h(w) = \frac{\exp(s_\theta(w, h))}{\sum_{w'} \exp(s_\theta(w', h))} \tag{2.1.21}$$

Noise-Contrastive Estimation

The softmax function quickly becomes a bottleneck, because it requires to compute $s_\theta(w', h)$ for all vocabulary items. Thus, the computational complexity increases linearly with the vocabulary size. In order to avoid this expensive calculation Mnih and Hinton, (2008) introduce a hierarchical LBL model to speed up the training in the same manner as the hierarchical CSLM does. They state however that it is not trivial to find an appropriate tree structure for the output layer. Hence, Mnih and Teh, (2012) and Mnih and Kavukcuoglu, (2013) propose to use Noise-Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2012). In NCE the unsupervised problem of density estimation is converted into a supervised binary classification problem. There, a classifier learns to distinguish between samples from the real data distribution P_d^h and samples from a noise distribution P_n. "In the language modeling setting, the data distribution $P_d^h(w)$ will be the distribution of words that occur after a particular context h" (Mnih and Teh, 2012). In other words, we try to fit the context-dependent model $P_\theta^h(w)$ to $P_d^h(w)$. Since the properties of the noise distribution are known, we can learn properties of the data distribution. Noise samples are considered to be k times more frequent than data samples, i.e., it is k times more likely that a sample w is drawn from the noise distribution. Given only the context h we can draw data and noise samples using

$$P(D = 1, w|h) = \frac{1}{k+1} P_d^h(w) \qquad (2.1.22)$$

$$P(D = 0, w) = \frac{k}{k+1} P_n(w). \qquad (2.1.23)$$

where $D = 1$ corresponds to the data distribution and $D = 0$ corresponds to the noise distribution.

The posterior probability that a word w came from the data distribution is given as:

$$P^h(D = 1|w;\theta) = \frac{P_\theta^h(w)}{P_\theta^h(w) + kP_n(w)} \qquad (2.1.24)$$

Instead of comparing the data sample to all items in the vocabulary it is now compared to only k noise samples. That makes the computation much faster. What is new about this method is that it sums up unnormalized and normalized probability distributions in the denominator. Thus, the algorithm must *learn to normalize the unnormalized distribution* $P_\theta^h(w)$ and therefore allow the binary classification. We drop the denominator in Equation 2.1.21 and directly use the unnormalized probability as $P_\theta^h(w)$. The overall objective function is:

$$\begin{aligned} J^h(\theta) = E_{P_d^h} &\left[\log P^h(D = 1|w;\theta)\right] \\ + kE_{P_n} &\left[\log P^h(D = 0|w;\theta)\right] \end{aligned} \qquad (2.1.25)$$

That means, we want to learn to distinguish between samples from the data distribution and samples from the noise distribution.

So far, we have only computed the objective function for a single context. We cannot train distributions of different contexts separately, because they share parameters, which are the word representations and the neural network parameters. In order to combine the per-context objective functions, the global NCE objective is given as a sum of the objectives of all contexts h

$$\begin{aligned} J(\theta) &= \sum_h P(h) J^h(\theta), \qquad (2.1.26) \\ &= E_{P(h)} J^h(\theta) \end{aligned}$$

where $P(h)$ are the empirical context probabilities.

In practice we can apply some simplifications. Since we know the correct word w in the given context h we can calculate the word's contribution to the overall objective by sampling k noise samples $x_1, \ldots, x_k$. For a context word pair this leads to

$$J^{hw}(\theta) = \log P^h(D = 1|w;\theta) + \sum_{i=1}^{k} \log P^h(D = 0|x_i;\theta) \tag{2.1.27}$$

We can see that we learn to distinguish between the real word and some noise samples.

Note, if we go over all windows in a corpus and calculate J^{hw} we do not need to apply Equation 2.1.26. The reason is that we get the weighting automatically, by seeing frequent contexts more often and therefore incorporate their cost more often. The overall objective is therefore

$$J(\theta) = \sum_{(hw)} J^{hw}(\theta) \tag{2.1.28}$$

As noise distribution Mnih and Teh, (2012) compare unigram and uniform distribution over the training vocabulary. They find that using a unigram distribution consistently gives better results. Using a more realistic distribution, such as a bi- or trigram distribution might improve the results even further or increase the training speed by requiring fewer noise samples (Mnih and Teh, 2012).

NCE only speeds up the model training. Normalization is still necessary during prediction. The normalized probability of word w for a context h is computed using the softmax function (Equation 2.1.21).

Vectorized Log-Bilinear Language Model

Mnih and Hinton, (2008) present another speedup technique of the LBL model. They set the position dependent weight matrices C_i to diagonal matrices, creating the vectorized Log-Bilinear Language model (vLBL). In this model the predicted word embedding (cf. Equation 2.1.18) is computed as:

$$\hat{\boldsymbol{q}}(h) = \sum_{i=1}^{n-1} \boldsymbol{c}_i \odot \boldsymbol{r}_{w_i} \tag{2.1.29}$$

, where $\boldsymbol{c}_i \in \mathbb{R}^d$ is the weight vector associated with position i in the context and $\odot$ is point-wise multiplication. The rest of the calculations stay the same.

2.1.5 *Training a Log-bilinear Language Model*

The standard way of training a LBL is by Stochastic Gradient Descent (SGD). In SGD a parameter θ_i is updated after every randomly sampled training example:

$$\theta_i \leftarrow \theta_i - \eta \frac{\partial J'}{\partial \theta_i} \tag{2.1.30}$$

where J' is the training objective, i.e., the cost function that is to be minimized. For the LBL the objective is $J' = J$. Often however, some regularization is wanted to avoid overfitting to the training data. This is added to the task-specific objective J. For ℓ_2 regularization that corresponds to:

$$J' = J + \sum_i \theta_i^2 \tag{2.1.31}$$

An alternative training strategy that can be used is AdaGrad (Duchi et al., 2011). In AdaGrad every parameter has its own learning rate. It gives "frequently occurring features very low learning rates and infrequent features high learning rates" (Duchi et al., 2011).

Let $g_{t,i} = \frac{\partial J'_t}{\partial \theta_i}$ be the gradient of parameter θ_i at time step t, then the parameter-specific learning rate $\eta_{t,i}$ is set to

$$\eta_{t,i} = \frac{\eta}{\sqrt{\sum_{t'=1}^{t} g_{t',i}^2}} \tag{2.1.32}$$

where η is the global learning rate. In other words, in AdaGrad we keep track of the squared gradients along the training process. Thus, the SGD update rule is rewritten as:

$$\theta_i \leftarrow \theta_i - \frac{\eta}{\sqrt{\sum_{t'=1}^{t} g_{t',i}^2}} g_{t,i} \tag{2.1.33}$$

The global learning rate η is less important in AdaGrad than it is in SGD, because it gets adjusted depending to the frequency with which a feature is seen (Dyer, 2013).

2.2 CONVOLUTIONAL NEURAL NETWORK

In this section we introduce Convolutional Neural Networks (CNNs) a classifier with very appealing properties for Sentiment Analysis (SA) as we will see below.

CNNs are a special type of feed-forward NN that originally were introduced for vision, more precisely handwritten digit recognition. In contrast to standard feed-forward Multi Layer Perceptron (MLP) which have a feature vector as input, a CNN directly uses the 2-dimensional image data as input (Denker et al., 1988; LeCun et al., 1989; 1990). The underlying idea is to find specific patterns in the input data independent of their exact position with fewer parameters than an MLP. This is achieved by having the following three appealing properties:

LOCAL (SPARSE) CONNECTIVITY One unit in a convolutional layer corresponds to a local neighborhood (i.e., a receptive field) in the input layer. That means that a single neuron is connected only to a *subset* of the units in the input layer (i.e., it is a *window* approach). This is in contrast to the usually fully connected MLP. The advantage of this approach is the sparser connectivity. Therefore, fewer parameters must be learnt.
SA benefits from this property, because most often, polarity is contained in only a few words of a sentence. The vast majority of words do not contribute to the polarity value and can therefore be ignored. Instead of looking at all the words in a sentence at ones, we can focus on windows containing the polarity.

SHARED WEIGHTS All units in a convolutional layer share the same weights, making it a *sliding window* approach. This also leads to fewer free parameters. However, the main advantage of this is that the same pattern is recognized independent of its position. For SA that means that the model recognizes feature combinations that indicate polarity, no matter where in the sentence they appear.

SUBSAMPLING / POOLING LAYERS The output of convolution layers is usually subsampled by pooling operations for two reasons: (i) The exact position of a certain pattern is usually not required to know. Instead, we just want to know roughly where a feature occurs (if at all). Thus, the exact position can be ignored. For SA this is beneficial, because we do not care about the position of the polarity indicator, but only about their polarity value. (ii) The output size is reduced even further, leading to fewer parameters in the following layers of the CNN.
Subsampling is either done on non-overlapping windows or on partially overlapping windows. Common pooling operations are average pooling LeCun et al., (1990) and max-pooling (Collobert and Weston, 2008).

In the following we describe all the building blocks of the popular CNN architecture *LeNet* for handwritten character recognition LeCun et al., (1998).

2.2.1 *Architecture*

A CNN receives a 2-dimensional input $Z \in \mathbb{R}^{n_1 \times n_2}$, where n_1 and n_2 are the dimensionalities of the input image. LeNet consists of three types of layers (in the following indicated by a superscript index): a convolution layer, a pooling layer, and a fully connected softmax layer.

2D Convolution

Using a convolution matrix $M \in \mathbb{R}^{m_1 \times m_2}$ (also called filter matrix) a CNN performs a 2d convolution that spans a region of size $m_1 \times m_2$:

$$A^{(1)}_{o,p} = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} M_{i,j} Z_{o+i,p+j} \tag{2.2.1}$$

where $A^{(1)}_{o,p}$ is the layer's activation at position $p \in [-n_1, n_1 - 1]$ and $o \in [-n_2, n_2 - 1]$. Positions outside the boundaries of Z are set to a default value (-1 in LeCun et al., (1989)). This approach makes sure that every row and column of the filter reaches every row and column of the input.

The output of the convolution is called *feature map* and has a size of $A^{(1)} \in \mathbb{R}^{(n_1+m_1-1)\times(n_2+m_2-1)}$. It is the input to a pooling layer.

Subsampling / Pooling

Pooling is used for further reducing the parameters of the model in the following layers and for achieving translation invariance of feature detectors. There are two common choices for pooling.[1]

Average pooling is defined as:

$$a^{(2)'} = \frac{1}{(n_1 + m_1 - 1) \times (n_2 + m_2 - 1)} \sum_o \sum_p A^{(1)}_{o,p} \tag{2.2.2}$$

Max pooling is defined as:

$$a^{(2)'} = \max_{o,p} A^{(1)}_{o,p} \tag{2.2.3}$$

The final output of the pooling layer is computed by adding a bias b and applying an element-wise non-linear activation function g:

$$a^{(2)} = g\left(a^{(2)'} + b^{(2)}\right) \tag{2.2.4}$$

A common choice for the activation function g is the hyperbolic tangent:

$$g(x) = tanh(x) \tag{2.2.5}$$

In order to detect multiple different patterns in the data, multiple filters exist in the network. Every filter has its own pooling step. Therefore, the output of the pooling layer is the combination of all pooled values:

$$\mathbf{a}^{(2)} = a^{(2)}_1 \dots a^{(2)}_f \tag{2.2.6}$$

[1] We assume a pooling over all values of the feature map, instead of a *pooling area*. This is not common when working with images. However, for the NLP applications in this work, using a different subsampling strategy is not necessary.

where f is the number of filters used.

In vision, usually there are multiple sequences of convolution and pooling layers, making it a deep network. Every convolution layer then recognizes more abstract features than the one before leading to the ability to detect complicated patterns.

Fully Connected Hidden Layers

The output of the last max-pooling layer is input to a sequence of fully-connected hidden layers. For simplicity we assume only one convolution layer, one pooling layer, and one fully-connected hidden layer:

$$\mathbf{a}^{(3)} = g\left(\mathbf{a}^{(2)}W^{(3)} + b^{(3)}\right) \tag{2.2.7}$$

with $W^{(3)}$ being the weight matrix of the hidden layer and $b^{(3)}$ being the bias of the hidden layer.

Softmax Layer

The final prediction is computed using another fully-connected layer with the softmax activation function:

$$\mathbf{a}_i^{(4)} = \frac{\exp(\mathbf{z}_i)}{\sum_j \exp(\mathbf{z}_j)} \tag{2.2.8}$$

with $\mathbf{z} = \mathbf{a}^{(3)}W^{(4)} + b^{(4)}$. The softmax converts the output into a proper probability distribution.

All parameters of the CNN $\theta = \{M^*, b^*, W^*\}$ are trained with SGD using back-propagation for computing the partial derivatives (LeCun et al., 1989; Rumelhart et al., 1986; Werbos, 1982).

2.2.2 *CNNs for NLP*

CNNs are especially powerful when dealing with 2-dimensional input. When we want to make use of the properties of CNNs in NLP, the question arises how to represent text. There are two main possibilities to represent a word in a NN model:

1. Distributional representations represent a word by a large vector containing cooccurrence statistics with all the other words in the vocabulary. The values in the vector can be (among others) simple cooccurrence counts (Bullinaria and Levy, 2007), tf-idf scores (Manning et al., 2009), or pointwise mutual information scores (Baroni et al., 2014; Bullinaria and Levy, 2007). The main advantage of this type of representation is the interpretability. Every dimension in the vector corresponds to one piece of information (usually the cooccurrence statistic). This however requires a long vector, because a single word is represented by the cooccurrence to *all* other words in the vocabulary.

2. Distributed representations – often called *word embeddings*, because they *embed* every word into a latent semantic space – on the other hand, distribute information among all available dimensions in a vector (Hinton, 1984; 1986). Additionally, a single dimension in such a vector participates in multiple pieces of information (Hinton et al., 1986). Such a representation "leads to automatic generalization" (Hinton, 1986). Additionally, the vectors are more compact and therefore smaller in dimensionality.

Both types of representations are based on the *distributional hypothesis*, which states that a word gets it semantics by the words it cooccurs with (Sahlgren, 2008). That means, if two words share the same set of other words they cooccur with, then their meaning must be similar (Karlgren and Sahlgren, 2001).

Baroni et al., (2014) compare several distributional representations with a distributed representation on a number of different tasks, such as semantic relatedness and synonymy detection. They show that models using distributional representations are superior to count models in many semantic tasks. While Levy et al., (2015) challenge this finding by stating that careful hyperparameter tuning makes both embeddings types performing similarly, they claim that skip-gram, one way of computing word embeddings, usually gives reasonable results and is the "fastest method to train, and cheapest (by far) in terms of disk space and memory consumption". In SA, models using word embeddings also perform very well (see e.g., dos Santos and Gatti, (2014), Kim, (2014), and Tang, Wei, Yang, et al., (2014)). Thus, we restrict our research to this type of word representations.

Word Embedding Methods

There are several popular alternative methods to compute distributed representations:

1. Early work created a cooccurrence matrix of words and applied a Singular Value Decomposition (SVD) on it (Schütze, 1992). Similarly, Lebret et al., (2013) applied a Hellinger Principal Component Analysis (PCA) on such a matrix.

2. As presented in Section 2.1 the CSLM from Bengio et al., (2000) learns word embeddings as a side product of the language modeling architecture. The major drawback of this method is the computational complexity, which makes learning of large vocabularies expensive. The same holds true for the LBL (Mnih and Hinton, 2007).

3. Mikolov, Chen, et al., (2013) presented two methods for using a NN model to learn semantic word vectors given an unlabeled text corpus, Continuous Bag-of-Words (CBOW) and skip-gram. In the CBOW model one tries to predict the word embedding

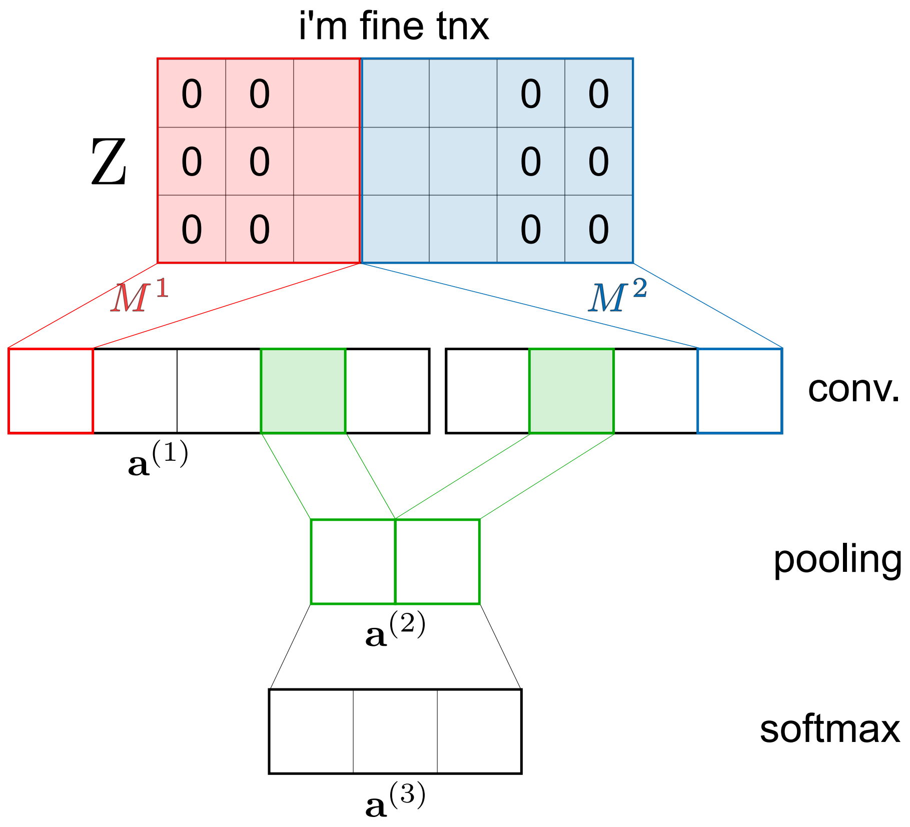


FIGURE 2.2.1: CNN ARCHITECTURE CNN architecture with embeddings layer, convolution, max-pooling, and the softmax.

of the target word from the sum of the word embeddings of the context words. In the skip-gram model one tries to predict the word embeddings of context words from the word embedding of a source word.

Both methods gained large popularity as initialization method for word representations in many NLP applications such as Morphology Induction (Soricut and Och, 2015), Named Entity Recognition (NER) (Passos et al., 2014), Part-of-Speech (POS) tagging (dos Santos and Zadrozny, 2014), and relation classification (dos Santos et al., 2015).

Due to their power and easy computation we use skip-gram to precompute word embeddings.

Word Embeddings in CNNs

Since we can create a 2-dimensional input for the CNN now using word embeddings, we can adapt the original model architecture. Let $LT \in \mathbb{R}^{d \times |V|}$ denote a lookup table that assigns each word in the vocabulary V a d-dimensional vector. This lookup table is trained with skip-gram (Mikolov, Chen, et al., 2013). The CNN architecture is depicted in Figure 2.2.1.

Given a sequence of n tokens t_1 to t_n the model concatenates all n word representations. The original input of the model Z therefore is replaced by:

$$Z = \begin{bmatrix} | & | & | \\ LT_{\cdot,t_1} & \cdots & LT_{\cdot,t_n} \\ | & | & | \end{bmatrix} \tag{2.2.9}$$

As stated before, word embeddings distribute information among all available dimensions, i.e., a single piece of information is not located in a single dimension. Therefore, we use filters that span all dimension to find features that interact with multiple dimensions: $M \in \mathbb{R}^{d \times m}$. The output of the first convolution layer is therefore a vector $\mathbf{a}^{(1)} \in \mathbb{R}^{(n+m-1)}$ instead of a matrix A as in Equation 2.2.1. The width of the filter m now specifies how many words the filter spans. Additionally, we can make use of multiple filter widths ($m \in \{3,4\}$ in Figure Figure 2.2.1) (Kim, 2014). This allows us to have filters that focus on shorter or longer patterns.

As before, the positions outside the boundaries of Z are set to a default value, which we set to zero. This is also known as *wide convolution* (Kalchbrenner et al., 2014). More precisely, we pad the input Z with $m-1$ zero columns at the left and right side (i.e., the sentence length becomes $n + 2 * (m-1)$).

Instead of neglecting all but one value per feature map in the pooling layer, Kalchbrenner et al., (2014) propose to use the k maximum values, which they call k-max pooling. Thus, more information about the data and its similarity to the filter is retained.

Let $\mathcal{N}_k$ be the set of the k largest values in the feature map $\mathbf{a}^{(1)}$, let π with $\pi_i \in [0, n+m-1]$ be a list of indices of the values in $\mathcal{N}_k$ sorted according to their indexes in the feature map $\mathbf{a}^{(1)}$ in ascending order. Then k-max pooling is defined as:

$$\mathbf{a}^{(2)'} = [\mathbf{a}^{(1)}_{\pi_i} | \forall 1 \leq i \leq k] \tag{2.2.10}$$

The activation function that is used in all our experiments in this work is the Rectified Linear Unit (ReLU) function :

$$g(x) = \max(0, x) \tag{2.2.11}$$

This non-linearity proved to be a crucial part in object recognition (Jarrett et al., 2009), MT (Vaswani et al., 2013), and ASR (Zeiler et al., 2013). It has some useful properties: (i) it is easier to optimize than other non-linear functions such as sigmoidal functions; (ii) it leads to faster convergence and better generalization; (iii) and is faster to compute (Zeiler et al., 2013).

We follow Kalchbrenner et al., (2014) and do not use a fully-connected hidden layer. Instead the output of the k-max pooling layer $\mathbf{a}^{(2)}$ is directly forwarded into the softmax layer (cf. Equation 2.2.8).

Please note that for English with its simple grammar, one layer of convolution and pooling is often sufficient, because there are rarely long-distance relations that make the interaction of filters at different positions necessary.

3 FINE-GRAINED CONTEXTUAL PREDICTIONS FOR HARD SENTIMENT WORDS

> This chapter covers work already published at international peer-reviewed conferences. The relevant publication is Ebert and Schütze, (2014). The research described in this chapter was carried out in its entirety by myself. The other author of the publication acted as advisor(s) or were responsible for work that was reported in the publication(s), but is not included in this chapter.

Many Sentiment Analysis (SA) systems use some kind of resource that indicates the polarity or even the valence of a word. Such resources mostly assign one polarity label (Hu and Liu, 2004) or one valence value to one word (Mohammad et al., 2013). Some sentiment lexicons such as the MPQA lexicon (Wilson et al., 2009) are more fine-grained and contain one polarity for different Part-of-Speech (POS) of a word. All these resources leave out information about the contextual usage of words. For instance "bright" in "bright mind" might be positive, whereas in "bright light" might be neutral. Since in both usages "bright" is an adjective, both occurrences would be labeled with the same polarity/valence. Therefore, the values in most available resources can only be considered a *prior* polarity/valence.

We put forward the hypothesis that high-accuracy SA is only possible if word senses with different polarity are accurately recognized. We provide evidence for this hypothesis in a case study for the word "hard" and propose Contextually Enhanced Sentiment Lexicons (CESLs) that contain the information necessary for sentiment-relevant sense disambiguation. An experimental evaluation demonstrates that senses with different polarity can be distinguished well using a combination of standard and novel features.

3.1 INTRODUCTION

This chapter deals with fine-grained sentiment analysis. We aim to make three contributions.

1. Based on a detailed linguistic analysis of contexts of the word "hard" (Section 3.2), we give evidence that highly accurate sentiment analysis is only possible if senses with different polarity are accurately recognized.

2. Based on this analysis, we propose to return to a lexicon-based approach to sentiment analysis that supports identifying sense

distinctions relevant to sentiment. Currently available sentiment lexicons give the polarity for each word or each sense, but this is of limited utility if senses cannot be automatically identified in context. We extend the lexicon-based approach by introducing the concept of a CESL. The lexicon entry of a word *w* in CESL has three components: (i) the senses of *w*; (ii) a sentiment annotation of each sense; (iii) a data structure that, given a context in which *w* occurs, allows to identify the sense of *w* used in that context.

As we will see in Section 3.2, the CESL sense inventory – (i) above – should be optimized for SA: closely related senses with the same sentiment should be merged whereas subtle semantic distinctions that give rise to different polarities should be distinguished.

The data structure in (iii) is a statistical classification model in the simplest case. We will give one other example for (iii) below: it can also be a set of centroids of context vector representations, with a mapping of these centroids to the senses.

3. If sentiment-relevant sense disambiguation is the first step in sentiment analysis, then powerful contextual features are necessary to support making fine-grained distinctions. Our third contribution is that we experiment with deep learning as a source of such features. We look at two types of deep learning features: word embeddings and neural network language model predictions (Section 3.3). We show that deep learning features significantly improve the accuracy of context-dependent polarity classification (Section 3.4) on a newly created dataset.

4. The newly created dataset with fine-grained sense labels is made publicly available.

This chapter is structured as follows. In Section 3.2, we present a linguistic analysis of different types of contexts of "hard" that are relevant for sentiment. Section 3.3 introduces our method: SA based on a CESL. Section 3.4 gives a preliminary experimental evaluation of CESL for the word "hard". Section 3.5 discusses related work. Section 3.6 presents our conclusions.

3.2 linguistic analysis of sentiment contexts of "hard"

For the linguistic analysis of the word "hard" we use the Amazon Product Review Data (Jindal and Liu, 2008). This dataset consists of about 5.8 million reviews taken from http://www.amazon.com. Every review concerns a product from one of the categories *books*, *music*, *DVD*, or *industry manufactured products*. Working with user reviews allows us to exploit a large variety of different contexts for "hard", because the speech is not restricted and therefore is informal and contains colloquial words.

From the 511 thousand contexts of "hard" in the dataset we took a random sample of 5000. 200 contexts are used a test set and another 200 are set aside for future use. We analyzed the remaining 4600 contexts using a tool we designed for this study, which provides functionality for selecting and sorting contexts, including a keyword in context display.

Our goal is to identify the different uses of "hard" that are relevant for sentiment. The basis for our inventory is the Cobuild (Sinclair, 1987) lexicon entry for "hard". We use Cobuild because it was compiled based on an empirical analysis of corpus data and is therefore more likely to satisfy the requirements of Natural Language Processing (NLP) applications than a traditional dictionary, such as WordNet.

Cobuild lists 16 senses. One of these senses (3) is split into two to distinguish the adverbial ("to accelerate hard") and adjectival ("hard acceleration") uses of "hard" in the meaning 'intense'. We conflated five senses (2, 4, 9, 10, 11) referring to different types of difficulty: "hard question" (2), "hard work" (4), "hard life" (11) and two variants of "hard on": "hard on someone" (9), "hard on something" (10). Another four different senses (3a, 5, 6, 7) referring to different types of intensity: "to work hard" (3a), "to look hard" (5), "to kick hard" (6), "to laugh hard" (7) are conflated as well. Furthermore, we identified a number of non-compositional meanings or phrases (lists NEGATIVE-P and NEUTRAL-P in the supplementary material[1]) in addition to the four listed by Cobuild (13, 14, 15, 16). Moreover, new senses for "hard" are introduced for opposites of senses of "soft": the opposite of 'quiet/gentle voice/sound' (7: music; e.g., "hard beat", "not too hard of a song") and the opposite of 'smooth surface/texture' (8: contrast; e.g., "hard line", "hard edge").

Table 3.2.1 lists the 10 different uses that are the result of our analysis. For each use, we give the corresponding Cobuild sense numbers, syntactic information, meaning, examples, typical patterns, polarity, and number of occurrences in our training and test sets. 7 of the identified uses are neutral and 3 are negative. However, in most sentiment lexicons, such as the MPQA (Wilson et al., 2009), "hard" is labeled as negative. The reason is that the vast majority of occurrences can be connected to the sense "difficult". This finding provides evidence for our hypothesis that senses need to be disambiguated to allow for fine-grained and accurate polarity recognition.

During the analysis, if a reliable pattern has been identified (e.g., the phrase "die hard" in Table 3.2.1), all contexts matching the pattern can be labeled automatically with the corresponding sense label. This way we create semi-automatic labels for the 4600 analyzed contexts. For the test set we hired two PhD students to label each of the 200 contexts with one of the 10 labels in Table 3.2.1. The inter-rater agreement Cohen's kappa is $\kappa = .78$. Disagreement was resolved by a third person.

[1] All supplementary material is available at http://www.cis.lmu.de/ebert.

We have published the labeled data set of 4600+200 contexts as supplementary material.

3.3 DEEP LEARNING FEATURES

We use two types of deep learning features to be able to make the fine-grained distinctions necessary for sense disambiguation.

1. We use word embeddings as features by averaging the embeddings of all words in the context (see below). This is similar to recent work, for instance Blacoe and Lapata, (2012).
2. We use a Neural Network Language Model (NNLM), the vectorized Log-Bilinear Language model (LBL) (Mnih and Kavukcuoglu, 2013), to predict the distribution of words for the position at which the word of interest occurs. For example, a Language Model (LM) will predict that words like "granite" and "concrete" are likely in the context "a * countertop" and that words like "serious" and "difficult" are likely in the context "a * problem". This is then the basis for distinguishing contexts in which "hard" is neutral (in the meaning 'firm, solid') from contexts in which it is a sentiment indicator (in the meaning 'difficult'). We will use the term Predicted Context Distribution (PCD) to refer to the distribution predicted by the LM.

LBL has three appealing features. (i) It learns state-of-the-art word embeddings (Mnih and Kavukcuoglu, 2013). (ii) The model is a language model and can be used to calculate PCDs. (iii) As a linear model, vectorized Log-Bilinear Language model (vLBL) can be trained much faster than other models, such as the original NNLM by Bengio et al., (2003).

3.4 EXPERIMENTS

The lexicon entry of "hard" in CESL consists of three components:

1. the senses
2. the polarity annotations (neutral or negative) and
3. the sense disambiguation data structure.

Components (i) and (ii) are shown in Table 3.2.1. In this section, we evaluate two different options for (iii) on the task of sentiment classification, namely classification and clustering.

3.4.1 *Classification*

The first approach is to use a statistical classification model as the sense disambiguation structure. The task given a context of "hard" is to pre-

		COBUILD				EXAMPLE			
	USE	SENSE	SYNTAX	MEANING	EXAMPLE	PATTERNS	SENTIMENT	# TRAIN	# TEST
1	firm	1	adjective	firm, stiff	hard floor, hard knot	hard N	neutral	87	5
2	difficult	1, 4, 9, 10, 11	adjective	difficult	hard question	hard for, hard on, hard to V	negative	2561	120
3	intense	3a, 5, 6, 7	adverb	intensely	work hard	V [so, too, as] hard, V hard	neutral	425	19
4	intense	3b	adjective	intense	hard look	be hard at it, hard N	neutral	24	7
5	hard man	8	adjective	unkind	hard man	hard man	negative	15	0
6	hard truth	12	attributive adjective	definitely true	hard truth	hard truth(s)	neutral	5	4
7	music	-	adjective	hard-rock-type music	hard beats	hard-rock(er)	neutral	347	15
8	contrast	-	adjective	opposite of soft transition	hard edge	hard edge(d), hard contrast	neutral	3	1
9	negative phrase	13, 15	phrases		hard drugs, hard to get	die hard	negative	36	2
10	neutral phrase	14, 16	phrases		hard disk	hard [copy, back, cover, bound]	neutral	375	27

TABLE 3.2.1: SENSE INVENTORY OF "HARD" These are all senses that are relevant for SA with their reference to the Cobuild sense number. Besides more detailed information, such as the syntactic structure, the meaning, concrete examples, example patterns, and the polarity for all senses are given. The last two columns show the number of occurrences of every sense in the training and test data.

			NGRAM	PCD	EMBED	ACC.	PREC.	REC.	F_1
DEVELOPMENT	BL	1				.62	.62	1.00	.76
	FULLY	2	+			.90	.91	.94	.92
		3		+		.90	.91	.92	.92
		4			+	.87	.87	.92	.90
		5	+	+		.92	.92	.94	.93
		6	+		+	.91	.90	.95	.92
		7		+	+	.86	.83	.96	.89
		8	+	+	+	.92	.93	.95	**.94**
	SEMI	9	+			.85	.87	.89	.88
		10		+		.85	.87	.89	.88
		11			+	.76	.73	.98	.83
		12	+	+		.85	.87	.89	.88
		13	+		+	.85	.87	.89	.88
		14		+	+	.85	.89	.87	.88
		15	+	+	+	.86	.87	.90	**.89**
TEST	BL	16				.66	.66	1.00	.80
	FULLY	17	+	+	+	.90	.89	.96	.92
	SEMI	18	+	+	+	.85	.85	.91	.88

TABLE 3.3.1: CONTEXT POLARITY RESULTS Results of the classification and clustering approaches using ngram, word embeddings, PCD, and their combinations as features. A "+" indicates that the feature type is active. "bl" denotes the majority baseline.

dict its polarity, either negative or neutral. We use liblinear (Fan et al., 2008) with standard parameters for classification based on three different feature types: ngrams, embeddings (embed) and PCDs. Ngram features are all n-grams for $n \in \{1, 2, 3\}$. As embedding features we use the concatenation of (i) the mean of the input space (R) embeddings and (ii) the mean of the target space (Q) embeddings of the words in the context as given by the LBL model. Blacoe and Lapata, (2012) showed that simply averaging word embeddings often yields better results than more complicated methods. As PCD features we use the PCD predicted by the LBL model for the sentiment word of interest, in our case "hard".

We split the set of 4600 contexts introduced in Section 3.2 into a training set of 4000 and a development set of 600. The contexts are prepared in a way that "hard" is the center word. All contexts are labeled as negative or neutral according to Table 3.2.1. We train the LBL model to receive the deep learning features, with stochastic gradient descent on mini-batches of size 100, following the Noise-Contrastive Estimation (NCE) training procedure of Mnih and Kavukcuoglu, (2013). We use AdaGrad (Duchi et al., 2011) with an initial learning rate set to $\eta = 0.5$. The embeddings size is set to 100.

We use a window size of $ws = 7$ for training the model. We found that the model did not capture enough contextual phenomena for $ws = 3$ and that results for $ws = 11$ did not have better quality than $ws = 7$, but had a negative impact on the training time. Using a vocabulary of the 100K most frequent words, we train the vLBL model for 4 epochs on 1.3 billion 7-grams randomly selected from the English Wikipedia.

Table 3.3.1 (lines 1–8) shows the classification results on the development set for all feature type combinations. Significant differences between results – computed using the approximate randomization test (Padó, 2006) – are given in Table 3.4.1. The majority baseline (bl), which assigns a negative label to all examples, reaches $F_1 = .76$. Our classifier is significantly better than the baseline for all feature combinations with F_1 ranging from .89 to .94. We obtain the best classification result (.94) when all three feature types are combined (significantly better than all other feature combinations except for 5).

3.4.2 *Clustering*

Manually labeling all occurrences of a word is expensive. As an alternative we investigate *clustering of the contexts of the word of interest*. Therefore, we represent each of the 4000 contexts of "hard" in the training set as its PCD,[2] use kmeans clustering with $k = 100$ and then label each cluster. This decreases the cost of labeling by an order of magnitude

[2] To transform vectors into a format that is more appropriate for the underlying Gaussian model of kmeans, we take the square root of each probability in the PCD vectors.

	1	2	3	4	5	6	7	8
1								
2	‡							
3	‡							
4	‡	‡	$*$					
5	‡			‡				
6	‡			‡				
7	‡	‡	†		‡	‡		
8	‡	†	†	‡		†	‡	

TABLE 3.4.1: SIGNIFIANCE Significant differences of lines 1–8 in Table 3.3.1. ‡: $p = 0.01$, †: $p = 0.05$, $*$: $p = 0.1$.

since only 100 clusters have to be labeled instead of 4000 training set contexts.

Table 3.3.1 (lines 9–15) shows results for this semi-supervised approach to classification, using the same classifier and the same feature types, but the cluster-based labels instead of manual labels for training.

For most feature combinations, F_1 drops compared to fully supervised classification. The best performing model for supervised classification (ngram+PCD+embed) loses 5%. This is not a large drop considering the savings in manual labeling effort. All results are significantly better than the baseline. There are no significant differences between the different feature sets (lines 9–15) with the exception of *embed*, which is significantly worse than the other 6 sets.

The centroids of the 100 clusters can serve as an alternative sense disambiguation structure for the lexicon entry of "hard" in CESL.[3] Each sense s is associated with the centroids of the clusters whose majority sense is s.

As final experiment (lines 16–18 in Table 3.3.1), we evaluate performance for the baseline and for PCD+ngram+embed – the best feature set – on the test set. On the test set, baseline performance is .80 (.04 higher than .76 on line 1, Table 3.3.1); F_1 of PCD+ngram+embed is .92 (.02 less than development set) for supervised classification and is .88 (.01 less) for semi-supervised classification (comparing to lines 8 and 15 in Table 3.3.1). Both results (.92 and .88) are significantly higher than the baseline (.80).

One thing to note is that the LBL model is trained on rather formal Wikipedia texts, whereas the contexts consists of user reviews, which are often informal and contain much colloquial speech. By using a different corpus, like a web corpus, the underlying word embeddings would reflect the word distribution of the "hard" contexts better and lead to improved classification results.

[3] The centroids are available as supplementary material.

3.5 RELATED WORK

Initial work on sentiment analysis was either based on sentiment lexicons that listed words as positive or negative sentiment indicators (e.g., Hu and Liu, (2004), Turney, (2002), and Yu and Hatzivassiloglou, (2003)), on statistical classification approaches that represent documents as ngrams (e.g., Pang et al., (2002)) or on a combination of both (e.g., Riloff, Wiebe, and Wilson, (2003), Whitelaw et al., (2005)). The underlying assumption of lexicon-based sentiment analysis is that a word always has the same sentiment, sometimes called *prior sentiment*. This is clearly wrong because words can have senses with different polarity, e.g., "hard copy" (neutral) vs. "hard memory" (negative).

Ngram approaches are also limited because ngram representations are not a good basis for relevant generalizations. For example, the neutral adverbial sense 'intense' of "hard" ("laugh hard", "try hard") vs. the negative adjectival meaning 'difficult' ("hard life", "hard memory") cannot be easily distinguished based on an ngram representation. Although ngram approaches could learn the polarity of these phrases they do not generalize to new phrases.

Wilson et al., (2005) present a more fine-grained polarity lexicon that contains polarity labels for POS-word pairs. The most similar sentiment lexicon to our work is SentiWordNet (Baccianella et al., 2010; Esuli and Sebastiani, 2006). It assigns 3 sentiment scores to each of the senses contained in WordNet (Miller, 1995). Although, this is a sense-based lexicon, the senses in WordNet are general and not focused on sentiment. As our analysis of "hard" shows, additional effort is necessary in order to adapt the available senses. Among the focus of our lexicon on the sentiment domain, we provide additional means to identify the polarity of a word in its context.

More recent compositional approaches to sentiment analysis can outperform lexicon and ngram-based methods (e.g., Socher et al., (2011), Socher et al., (2013)). However, these approaches conflate two different types of contextual effects: differences in sense or lexical meaning ("hard memory" vs. "hard wood") on the one hand and meaning composition like negation on the other hand. From the point of view of linguistic theory, these are different types of contextual effects that should not be conflated. Recognizing that "hard" occurs in the scope of negation is of no use if the basic polarity of the contextually evoked sense of "hard" (e.g., negative in "no hard memories" vs. neutral in "no hard wood") is not recognized.

Wilson et al., (2009) present an approach to classify contextual polarity building on a two-step process. First, they classify if a sentiment word is polar in a phrase and if so, second, they classify its polarity. Our approach can be seen as an extension of this approach; the main difference is that we show in our analysis of "hard" that the polarity of phrases depends on the senses of the words that are used. This is evidence that high-accuracy polarity classification depends on sense disambiguation.

There has been previous work on assigning polarity values to senses of words taken from WordNet (e.g., Baccianella et al., (2010), Wiebe and Mihalcea, (2006)). These approaches again can be considered a prior polarity and therefore do not disambiguate the sense of a word given its context.

Akkaya et al., (2009) introduce subjectivity word sense disambiguation, "which is to automatically determine which word instances in a corpus are being used with subjective senses, and which are being used with objective senses". The authors propose a system that uses one classifier for each word in a lexicon. These classifiers classify every occurrence of the respective word as being subjective or objective. Several existing subjective/objective classifiers were adapted to the new approach and showed superior performance. Additionally, the proposed subjective/objective classifier was used as preprocessing step in polarity classification. Our classification procedure directly classifies polarity instead of subjectivity and objectivity. As Scheible and Schütze, (2013) show, subjectivity classification is not sufficient for sentiment.

Previous work on representation learning for sentiment analysis includes Maas and Ng, (2010) and Maas et al., (2011). Their models learn word embeddings that capture semantic similarities and word sentiment at the same time. Their approach focuses on sentiment of entire sentences or documents and does not consider each sentiment word instance at a local level.

We present experiments with one supervised and one semi-supervised approach to Word Sense Disambiguation (WSD) in this chapter. Other WSD approaches, e.g., thesaurus-based WSD (Yarowsky, 1992), could also be used for CESL.

3.6 CONCLUSION

The sentiment of a sentence or document is the output of a causal chain that involves complex linguistic processes like contextual modification and negation. Our hypothesis in this chapter was that for high-accuracy sentiment analysis, we need to model the root causes of this causal chain: the meanings of individual words. This is in contrast to other work in sentiment analysis that conflates different linguistic phenomena (word sense ambiguity, contextual effects, negation) and attempts to address all of them with a single model.

For sense disambiguation, the first step in the causal chain of generating sentiment, we proposed CESL, a Contextually Enhanced Sentiment Lexicon that for each word w holds the inventory of senses of w, polarity annotations of these senses and a data structure for assigning contexts of w to the senses. We introduced new features for sentiment analysis to be able to perform the fine-grained modeling of context needed for CESL. In a case study for the word "hard", we showed that high accuracy in sentiment disambiguation can be achieved using our approach.

All supplementary material is available at http://www.cis.lmu.de/ebert.

3.7 FUTURE WORK

Possible extensions of our work are:

- In this chapter only a single word is analyzed. Thus, it needs to be shown that our findings generalize to the entire sentiment lexicon.

- Although the presented clustering method reduces manual labeling effort, it is still a time consuming process. One possible approach is to search for words in the contexts that are uniquely associated with one sense, such as "die hard" and then extend the pattern by synonyms of the found word using again a manual lexicon (e.g., WordNet (Miller, 1995)). Yarowsky, (1995) proposes a bootstrapping approach for generating a growing training set by starting with high quality seed patterns, then training a classifier and using those newly found patterns for the training set in the next iteration that receive a high classifier confidence.

- A more sophisticated approach for computing a context representation out of the single word embeddings can further improve the performance of out system. Dinu et al., (2013) give a starting point by comparing several methods.

4 LINEAR VERSUS NON-LINEAR LANGUAGE MODELS

In the previous chapter we use a Language Model (LM) to improve contextual polarity classification. The model of choice is the Log-Bilinear Language model (LBL). Since it is a linear model, it has a serious drawback compared to non-linear models. Words do not interact with each other in a linear way. For example, "her" can either be a personal pronoun as in "to see her" or a possessive pronoun as in "her book". The meaning of "her" hence depends on the words in the context. Such a non-linear behavior can only limitedly be handled by a linear model.

Moreover, since a sequence of linear layers can be replaced by one big layer, non-linear layers are building bricks for deep neural networks. Deep non-linear neural networks are more powerful and compact than shallow or less deep architectures (Bengio, 2009). The success of deep neural networks in various Machine Learning (ML) tasks proves that.

In this chapter we introduce a very simple extension to the LBL and show in an empirical parameter study that even this shallow model can benefit from a non-linearity. Due to its simplicity, the proposed method is easily applicable to other linear models, such as Mikolov's popular Continuous Bag-of-Words (CBOW) model (Mikolov, Chen, et al., 2013).

4.1 INTRODUCTION

Language modeling, the task of assigning a probability to a sequence of words, is a crucial task for many applications in Natural Language Processing (NLP), such as Machine Translation (MT) (e.g., Vaswani et al., (2013)), Automatic Speech Recognition (ASR) (e.g., Schwenk and Gauvain, (2005) and Schwenk, (2007)), or Sentiment Analysis (SA) (Ebert and Schütze, 2014). Since their introduction, Neural Network Language Model (NNLM) (Bengio et al., 2003) have received much attention. One reason for their success is the way of learning word representations. The model embeds all words into a low-dimensional vector space and automatically learns similar word embeddings for words that occur in similar contexts. Therefore, the embeddings of "Monday" and "Tuesday" will be similar, whereas the embeddings of "Monday" and "love" will be dissimilar.

Combining several word embeddings in NNLMs addresses the sparsity problem of ngram models, because word sequences do not have to be memorized but are combined by the neural network. Thus, the next word can always be predicted, even for a previously unseen contexts, which for a standard ngram model can only be achieved with techniques such as smooting (Goodman, 2001) or back-off (Katz, 1987). Unfortunately, training an NNLM is computationally expensive, because in the final prediction step a normalization over the entire vocabulary is necessary, to receive a probability distribution.

A more efficient continuous space LM is the LBL model (Mnih and Hinton, 2007). It is a linear model, which can be trained efficiently, yields comparable results to NNLM (Mikolov et al., 2011) and converges faster (Le et al., 2010). However, Bengio, (2009) argues that deep *non-linear* architectures have "greater expressive power" than their shallow linear counterparts. Therefore, in this chapter we show that a non-linearity even helps in a shallow LBL model and make the following contributions:

1. We show that adding a non-linearity is useful when a model has only a limited number of parameters. This can happen, when not enough training data is available to train a full-size model.

2. To prove that, we introduce simple non-linear extensions to the LBL and the vectorized Log-Bilinear Language model (vLBL). Our method can be applied to other linear models as well (Section 4.2).

3. We perform an exhaustive empirical parameter study that shows under which circumstances non-linear models are better than linear models. We show that the linear model is sufficient for language modeling tasks in which – given the corpus and vocabulary sizes – a sufficient number of parameters is available to handle the data's variety. The more difficult the task becomes, e.g., by having fewer embeddings dimensions, the more important non-linearity becomes (Table 4.3.2).

4. Finally, in Table 4.3.2 we show that non-linear models are less correlated with a modified Kneser-Ney (KN) model and therefore benefit more from interpolation than the linear models.

This Chapter is structured as follows. Section 4.2 introduces our extensions to the LBL model. In Section 4.3 we present the empirical parameter study and show results with all 3- and 7-gram model variants as well as interpolated results with a modified KN model. The following Section (Section 4.4), presents related work. The chapter is concluded in Section 4.5 and Section 4.6 suggests future work.

4.2 non-linear lbl variants

Both variants of the model LBL and vLBL are linear models in that they linearly combine weights and word embeddings to predict the next word. Thus, interaction of words with each other is also limited to be linear. In order to increase the power of the models we propose to add a non-linear function f to the predicted word representation of the LBL model (cf. Equation 2.1.18) creating a non-linear matrix model (non-linear Log-Bilinear Language model (nLBL)):

$$\hat{\boldsymbol{q}}_{nLBL} = f\left(\hat{\boldsymbol{q}}_{LBL}\right) \tag{4.2.1}$$

We can do the same with the vLBL model (cf. Equation 2.1.29, non-linear vectorized Log-Bilinear Language model (nvLBL)):

$$\hat{\boldsymbol{q}}_{nvLBL} = f\left(\hat{\boldsymbol{q}}_{vLBL}\right) \tag{4.2.2}$$

As non-linear function we use a Rectified Linear Unit (ReLU) (cf. Equation 2.2.11) (Nair and Hinton, 2010) In preliminary experiments it showed improved results over the usually used *tanh* function.

4.3 EXPERIMENTS

We perform an exhaustive empirical parameter analysis on the Wall Street Journal (WSJ) part of the Penn Treebank. For comparability reasons we use the preprocessed version from Mikolov et al., (2010). It contains 930K/74K/82K tokens in the training/development/test set. The vocabulary consists of the 10K most frequent word types. Other types are mapped to the unknown token. The preprocessing includes tokenization, lowercasing, converting numbers to the generic symbol "N", and the removal of punctuation that is not part of an abbreviation (e.g., the period in "ms." and the apostrophe in "do n't" are retained).

We first report results for 3-gram models – i.e., train a model that predicts a word given its two preceding words – as this is a standard model in many applications. Since we want to improve the LM as part of our "hard" classification pipeline (cf. Chapter 3) we additionally again use a 7-gram model. As noise distribution for Noise-Contrastive Estimation (NCE) the unigram distribution of words in the training set is used, and the number of noise samples is set to $k = 5$. Note that a higher number of noise samples would lead to better performance but longer training times. We train all models with mini-batch Stochastic Gradient Descent (SGD), having a mini-batch size of 100 examples. The training is stopped if 100 epochs of training are finished or five consecutive epochs lead to decreasing performance (early stopping), which ever comes first. We use AdaGrad (Duchi et al., 2011) for dynamic learning rate adjustment and make use of ℓ_2 regularization.

To find reasonable parameter choices that we can then use in the grid search, we trained vLBL models with several parameter configurations for some epochs. The parameters in Table 4.3.1 lead to reasonable performances and are thus used in a grid search. Combining all parameter configurations leads to 240 configurations for each of the 4 model types. All experiments are evaluated intrinsically by computing Perplexity (PPL).

PARAMETER	DESCRIPTION	VALUES
$\mathrm{lr}_{\mathrm{emb}}$	learning rate of word embeddings	$\{1, 10^{-1}, 10^{-2}\}$
$\mathrm{lr}_{\mathrm{default}}$	learning rate of remaining parameters (e.g., position dependent weights)	$\{1, 10^{-1}, 10^{-2}, 10^{-3}\}$
λ	weight decay (ℓ_2 regularization)	$\{10^{-5}, 10^{-6}\}$
m	word embeddings size	$\{10, 20, \ldots, 100\}$

TABLE 4.3.1: ANALYZED PARAMETERS List of parameters that are analyzed with a description and their value ranges.

MODEL	$\mathrm{lr}_{\mathrm{emb}}$	$\mathrm{lr}_{\mathrm{default}}$	λ
vLBL	0.01	0.1	10^{-5}
NVLBL	0.01	0.1	10^{-5}
LBL	0.1	0.001	10^{-6}
NLBL	1.0	1.0	10^{-5}

TABLE 4.3.2: HYPERPARAMETERS OF THE BEST 3-GRAM MODELS Best parameter configuration for $m = 100$ embeddings dimensions, according to the development set performance.

4.3.1 *Results 3-gram*

Table 4.3.2 lists the parameters for each model type that led to the lowest perplexity for 100 embeddings dimensions on the development set. While the vectorized models yield the best performance with the same parameter configuration, the parameters for the matrix models need to be quite different in order to reach the best performance.

In Figure 4.3.1 and Table 4.3.3 we can see that the NVLBL model is considerable worse than the other models with only 10 embeddings dimensions. However, it can catch up with 20-30 dimensions. Between 20 and 60 dimensions, the models reach similar performance, with the NLBL model always being the best. With more than 60 dimensions, the LBL model cannot improve anymore. Our analysis suggests that this is a problem of overfitting of the LBL model due to the larger number of parameters. Thus, it is even more surprising that the non-linear matrix model (NLBL) does not seem to be susceptible to this problem and reaches the best development set performance with 176.8 PPL. However, the difference between the models is almost neglectible.

The left half of Table 4.3.4 shows the performance on the development and test set of the WSJ data of the single best model per model type that make use of 100 dimensional embeddings. The test set seems

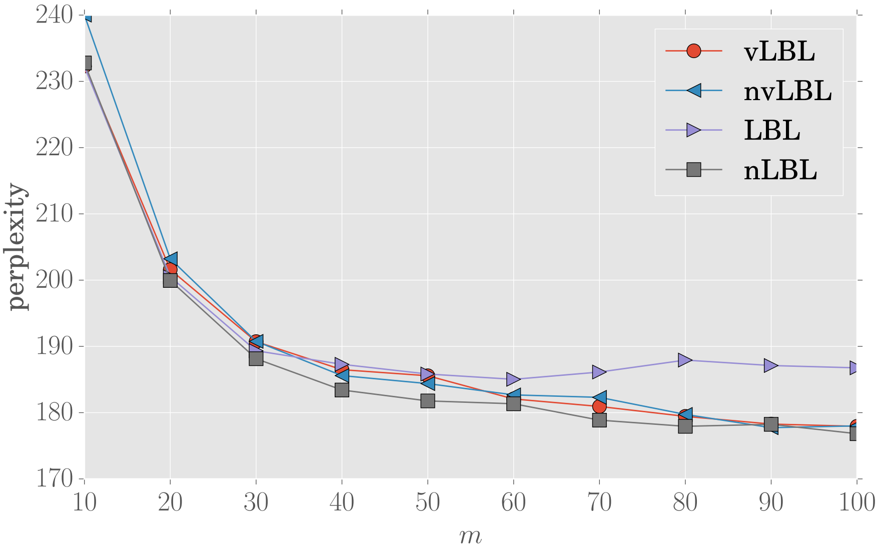


FIGURE 4.3.1: PERPLEXITY OF 3-GRAM MODELS PER WORD EMBEDDINGS SIZE Best PPL performances of all 3-gram models per embeddings size m on the development set.

m	vLBL	nvLBL	LBL	nLBL
10	232.3	240.0	**232.2**	232.8
20	201.6	203.2	200.5	**199.9**
30	190.7	190.8	189.3	**188.1**
40	186.5	185.6	187.3	**183.4**
50	185.6	184.4	185.8	**181.8**
60	182.1	182.7	185.0	**181.3**
70	180.9	182.3	186.1	**178.9**
80	179.4	179.7	187.9	**177.9**
90	178.3	**177.7**	187.1	178.2
100	177.9	178.0	186.8	**176.8**

TABLE 4.3.3: PERPLEXITY OF 3-GRAM MODELS PER WORD EMBEDDINGS SIZE Best PPL performances of all 3-gram models per embeddings size m on the development set. Bold is best per row.

	SINGLE		INTERPOLATED	
MODEL	DEV	TEST	DEV	TEST
vLBL	177.9	166.6	140.7	132.8
NVLBL	178.0	**164.9**	140.9	132.2
LBL	186.8	174.7	141.9	133.9
NLBL	**176.8**	165.1	**137.8**	**129.9**

TABLE 4.3.4: RESULTS OF BEST 3-GRAM MODELS Results of the best 3-gram models with 100 dimensional word embeddings. *single* denotes the performance of the LBL models alone. *interpolated* is the performance of the LBL models interpolated with a modified KN model. Bold is the best perplexity per column.

to be more similar to the training set than the development set, because all models reach lower perplexity on it. While the non-linear matrix model yields the lowest PPL on the development set, the NVLBL model takes over on the test set. The difference between development and test set performance is similar for all models, indicating that no model is better or worse in generalization than another. As can be seen, the non-linear variant of the vector model improves on the linear variant only on the test set. For the matrix models, the difference between linear and non-linear models is larger.

Interpolation

Now we interpolate all our models with a modified KN 3-gram model, as estimated by the srilm toolkit.[1] As before, only the best parameter configurations are reported. To simplify the procedure we equally weight KN and LBL models. Figure 4.3.2 and Table 4.3.5 shows the results. The KN model alone yields 157.8 PPL on the development set and 148.3 on the test set.

Interpolating is beneficial for all four LBL model types. However, the perplexity for both vector models stagnates at about 80 embeddings dimensions. The performance of the non-linear matrix model keeps on becoming lower and reaches its lowest perplexity of 137.8 at 100 embeddings dimensions. We conclude that the LBL model learns features that are more orthogonal to the KN model and can therefore benefit more from the interpolation. The performance of the best development set models per model type (left half of Table 4.3.4), interpolated with the KN model on the development and test set are listed in the right half of Table 4.3.4.

[1] http://www.speech.sri.com/projects/srilm/

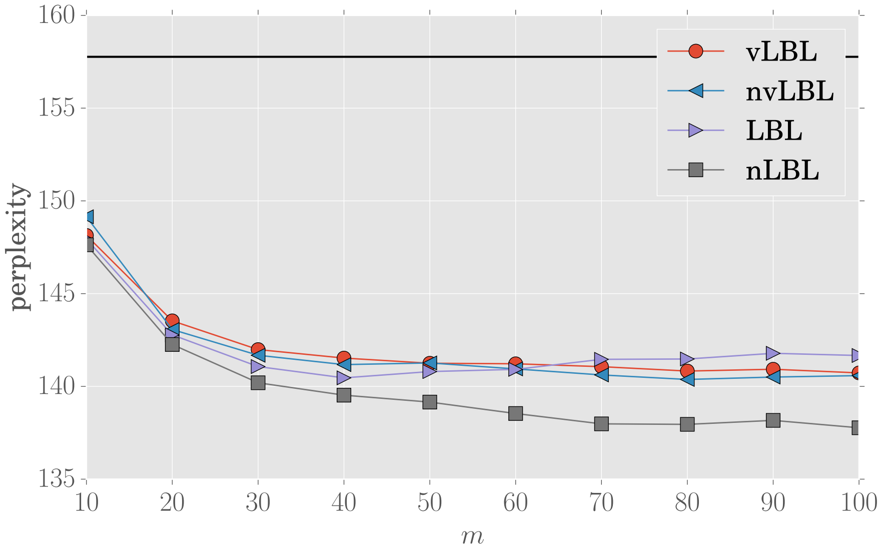


figure 4.3.2: interpolated perplexity of 3-gram models per word embeddings size Best interpolated PPL performances of all 3-gram models per embeddings size m on the development set. The solid line corresponds to the 3-gram KN model.

m	vLBL	nvLBL	LBL	nLBL
10	148.1	149.1	147.9	**147.6**
20	143.5	143.1	142.8	**142.3**
30	142.0	141.7	141.1	**140.2**
40	141.5	141.2	140.5	**139.5**
50	141.2	141.3	140.8	**139.2**
60	141.2	140.9	140.9	**138.5**
70	141.1	140.6	141.5	**138.0**
80	140.8	140.4	141.5	**138.0**
90	140.9	140.5	141.8	**138.2**
100	140.7	140.6	141.7	**137.8**

table 4.3.5: interpolated perplexity of 3-gram models per word embeddings size Best interpolated PPL performances of all 3-gram models per embeddings size m on the development set. Bold is best per row.

MODEL	lr_{emb}	$\text{lr}_{\text{default}}$	λ
vLBL	0.01	1.0	10^{-6}
NVLBL	0.01	1.0	10^{-6}
LBL	0.1	0.1	10^{-5}
NLBL	1.0	1.0	10^{-5}

TABLE 4.3.6: HYPERPARAMETERS OF THE BEST 7-GRAM MODELS Best parameter configuration for $m = 100$ embeddings dimensions, according to development set performance.

4.3.2 *Results 7-gram*

We now repeat all experiments for 7-gram models, because we use these models in Chapter 3. Table 4.3.6 lists the parameter configurations for all models yielding the lowest perplexity for 100 dimensional word embeddings on the development set. As before, both vector models require equal parameters, whereas both matrix models perform best with different parameter configurations.

Figure 4.3.3 and Table 4.3.7 depict the perplexities of the best parameter configuration of every model type for all embeddings sizes on the development set. For the setting of $m = 10$, the choice of model is not important, since no model is able to capture sufficient variation, due to the small number of parameters. The vLBL model yields better performance than the NVLBL model with $m \leq 30$. With more dimensions, the NVLBL is consistently better. That shows that having a non-linearity, even such a simple one, is beneficial.

The linear matrix model (LBL) performs much worse than the other models starting from $m = 60$ (similar to the 3-gram experiments). This again seems to be caused by overfitting of the LBL model due to the larger number of parameters. Thus, it is even more surprising that the non-linear matrix model (NLBL) does not seem to be susceptible to that problem and reaches the best development set performance with 162.6 PPL.

Another explanation for LBL's low performance might be the parameter choice. It is possible that the LBL model needs different parameters than the ones presented in Section 4.3 to find better minima. The right choice might be more crucial for a model having more parameters than the vector models.

The best single model's performance for 100 dimensions on the development and test set are listed in the left part of Table 4.3.8. Again, the test set PPLs are better than the development set PPLs, as all models reach lower perplexity. Interestingly, the vector-based models increase the distance to their matrix-based counterparts. This result indicates that the vector-based models are better in generalizing to unseen data.

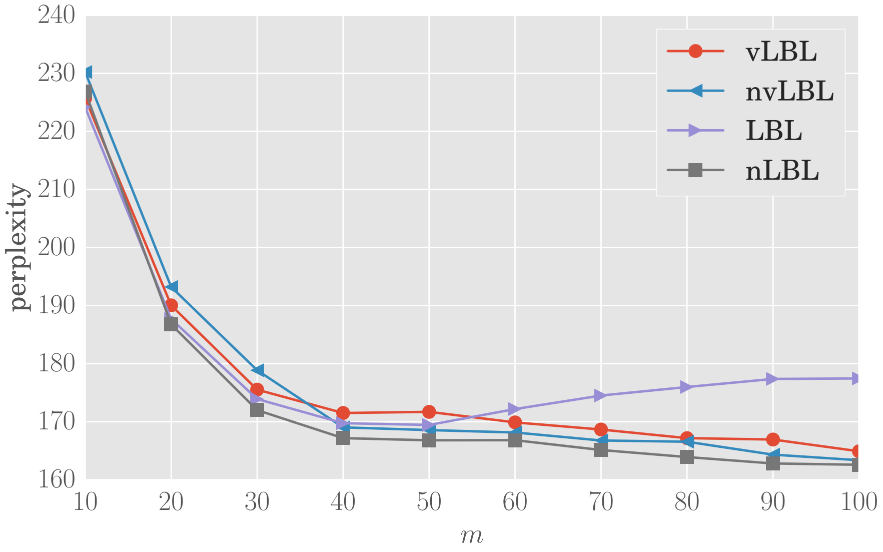


FIGURE 4.3.3: PERPLEXITY OF 7-GRAM MODELS PER WORD EMBEDDINGS SIZE Best PPL performances of all 7-gram models per embeddings size m on the development set.

m	vLBL	nvLBL	LBL	nLBL
10	225.7	230.2	**224.2**	226.9
20	190.0	193.2	187.7	**186.8**
30	175.5	178.9	173.9	**172.0**
40	171.5	169.0	169.8	**167.2**
50	171.7	168.6	169.4	**166.8**
60	169.9	168.1	172.2	**166.8**
70	168.7	166.8	174.5	**165.1**
80	167.2	166.6	176.0	**163.9**
90	166.9	164.3	177.4	**162.8**
100	164.9	163.4	177.5	**162.6**

TABLE 4.3.7: PERPLEXITY OF 7-GRAM MODELS PER WORD EMBEDDINGS SIZE Best PPL performances of all 7-gram models per embeddings size m on the development set. Bold is best per row.

	single		interpolated	
model	dev	test	dev	test
vLBL	164.9	153.7	124.8	119.0
nvLBL	163.4	**152.6**	123.8	118.1
LBL	177.5	170.9	126.2	122.0
nLBL	**162.6**	154.8	**121.4**	**116.7**

table 4.3.8: results of best 7-gram models Results of the best 7-gram models with 100 dimensional word embeddings. *single* denotes the performance of the LBL models alone. *interpolated* is the performance of the LBL models interpolated with a modified KN model. Bold is the best perplexity per column.

On the test set, both non-linear models perform better than their linear counterpart. However, the difference is rather small.

Please note that Mikolov et al., (2011) report a perplexity of 144.5 for the LBL model, compared to our result of 170.9. Since no details about the model parameters are given we tried several parameter configurations to match the reported performance. We reached 145.7 PPL with an 11-gram model. Besides the context size another difference between our and their implementation is that our training uses NCE, which might perform a little worse than the maximum likelihood training.

Interpolation

The 7-gram KN model alone reaches 147.4 PPL on the development and 140.9 on the test set. Interpolating with a 7-gram modified KN model is beneficial for all four LBL model types. vLBL reaches its lowest perplexity at 70 dimensions, while the non-linear version of it further reduces perplexity up to 100 dimensions. The performance of the non-linear matrix model keeps on becoming lower and reaches its lowest perplexity of 121.4 at 100 embeddings dimensions. We conclude that the LBL model learns features that are more orthogonal to the KN model and can therefore benefit more from the interpolation. The performance of the best model per model type with 100 dimensional embeddings on the development set (left half in Table 4.3.8) interpolated with the 7-gram KN model is listed in the right part of Table 4.3.8.

4.4 related work

NNLMs' major problem is the long training time. Several speed-up techniques have been proposed to make the training of large NNLMs feasible. They include short lists (Schwenk, 2004), class-based predic-

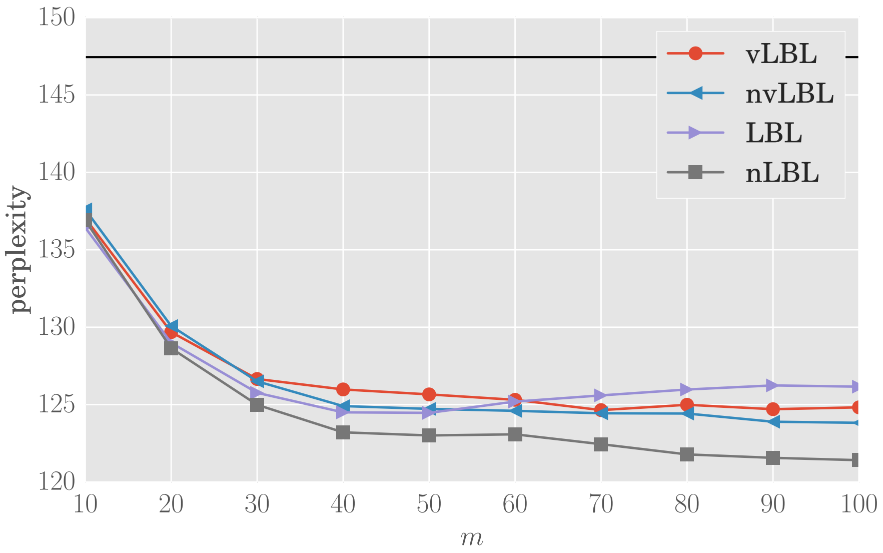


FIGURE 4.3.4: INTERPOLATED PERPLEXITY OF 7-GRAM MODELS PER WORD EMBEDDINGS SIZE Best interpolated PPL performances of all 3-gram models per embeddings size m on the development set. The solid line corresponds to the 3-gram KN model.

m	vLBL	nvLBL	LBL	nLBL
10	137.0	137.6	**136.4**	136.9
20	129.7	130.1	129.0	**128.6**
30	126.7	126.5	125.8	**125.0**
40	126.0	124.9	124.5	**123.2**
50	125.7	124.7	124.5	**123.0**
60	125.3	124.6	125.2	**123.1**
70	124.6	124.4	125.6	**122.4**
80	125.0	124.4	126.0	**121.8**
90	124.7	123.9	126.2	**121.6**
100	124.8	123.8	126.2	**121.4**

TABLE 4.3.9: INTERPOLATED PERPLEXITY OF 7-GRAM MODELS PER WORD EMBEDDINGS SIZE Best interpolated PPL performances of all 3-gram models per embeddings size m on the development set. Bold is best per row.

tion (Goodman, 2001; Le et al., 2013), the usage of mini-batches and SGD, the use of optimized BLAS libraries and early stopping (Bengio et al., 2003; Schwenk, 2004). Although these techniques improve the training speed of NNLMs, the LBL model can also benefit from all these methods (e.g., for class-based prediction in LBL see Botha and Blunsom, (2014)) and is therefore even more efficient to train.

The prominent skip-gram (Mikolov, Chen, et al., 2013) model could be seen as an alternative to the LBL, but it lacks the capability of ordering information. It behaves like a bag-of-words model and doesn't distinguish between "dog bites man" and "man bites dog". Therefore, it is not suited for LM tasks. Recently, and after our study was finished, Ling et al., (2015) proposed an extensions to the skip-gram model that makes it position-dependent. They show improvements of their model on two syntactic tasks, Part-of-Speech (POS) tagging and dependency parsing.

Little research has been conducted on the comparison of linear and non-linear models. M. Wang and Manning, (2013) compared linear and non-linear feed forward networks on Named Entity Recognition (NER) and chunking tasks. They found that non-linear architectures lead to better performance in low-dimensional space, whereas linear architectures perform equally well (and even a little bit better in some tasks) in a high dimensional discrete feature space. Lebret et al., (2013) report diverse results with respect to linearity. They show that linear networks using different types of word representation as input yield similar results to non-linear ones on polarity classification. However, in NER non-linear networks are better. Arisoy et al., (2012) compared a shallow NNLM with a deep NNLM with the same number of parameters and found that the deep NNLM yields better results. This finding suggests that non-linear functions might be crucial. In this chapter we show that non-linear versions of the LBL model lead to better performance in language modeling.

4.5 CONCLUSION

In this chapter we have introduced a very simple non-linear extension of the popular LBL model and have shown that it reaches lower perplexity than the original model. This is especially true, if there are too few parameters to cope with the data's variety. Therefore, this is helpful in scenarios where large models cannot be learnt sufficiently, e.g., when there is not enough training data. Thus, we suggest to always use the non-linear version, because the computational overhead is minimal, but it can lead to superior performance. We have further shown that the vectorized non-linear model is better than the matrix-based model when there is no interpolation. In the case of interpolation with a KN model, the matrix model seems to learn more orthogonal features, because it benefits more from the ngram model.

4.6 future work

There are possible directions for future work:

- The presented extension is very simply and can be replaced by a more powerful version. For instance, $\hat{\boldsymbol{q}}_{nvLBL}$ can be computed as:

$$\hat{\boldsymbol{q}}'_{nvLBL} = \sum_{i=1}^{n-1} f\left(\boldsymbol{c}_i \odot \boldsymbol{r}_{w_i}\right)$$

- Due to its relatively small size, the WSJ corpus is well suited for exhaustive parameter studies as this one. As next step, the findings in this chapter need to be verified on a larger corpus, such as the APNews corpus (Bengio et al., 2003) or the One Billion Word corpus (Chelba et al., 2013).

5 LINGUISTICALLY-INFORMED CONVOLUTIONAL NEURAL NETWORKS

> This chapter covers work already published at international peer-reviewed conferences. The relevant publications are Ebert et al., (2015b) and Ebert et al., (2015a). The research described in this chapter was carried out in its entirety by myself. The other author(s) of the publication(s) acted as advisor(s) or were responsible for work that was reported in the publication(s), but is not included in this chapter.

As we saw in Chapter 3, fine-grained Sentiment Analysis (SA) requires a substantial amount of manual work. Moreover, in many cases labels on sub-sentence levels may not even be required, e.g., when one is interested in the polarity of entire reviews. In an ideal case we want a classifier to automatically make these distinctions. Therefore, in this chapter we will improve on a standard architecture, a Convolutional Neural Network (CNN) for polarity classification of entire pieces of text, such as whole sentences or in our case Twitter tweets. The model is supposed to learn fine-grained interactions using prior polarity and contexts. A CNN is more suited for this task than the Language Model (LM) approach we followed in Chapter 3, because it is a sequence model that analyzes the entire text instead of only a window around a given word. Thus, also long-distance dependencies can be learnt. Additionally, CNNs focus on only the most salient features of the input. This is a beneficial property in SA, because the polarity of a text very often is determined by a couple of words only. Having multiple feature detectors allows for resolving long-distance relationships such as negation.

A standard CNN has no information about polarity or knowledge of positivity or negativity. But research has shown that linguistic knowledge in terms of sentiment lexicons and other linguistic resources proved to be beneficial in polarity classification. This chapter introduces a linguistically-informed Convolutional Neural Network (lingCNN), which incorporates this valuable kind of information into the model. We present two intuitive and simple methods: The first method integrates word-level features, the second sentence-level features. By combining both types of features our model achieves results that are comparable to state-of-the-art systems.

5.1 INTRODUCTION

This chapter explores the use of CNNs for SA. CNNs reach state-of-the-art results in several polarity classification tasks (Kalchbrenner et al., 2014; Kim, 2014; Severyn and Moschitti, 2015; Tang, Wei, Qin, Liu, et al., 2014). Reasons are their ability to deal with arbitrary input sentence lengths and to preserve word order. Moreover, they learn to find the most important polarity indicators and ignore the rest of the sentence. That is beneficial, since most of the words in a text do not convey sentiment information. Finally, CNNs can make use of powerful pre-trained word representations (e.g., Mikolov, Chen, et al., (2013)).

Despite its power, a CNN does not know about sentiment and therefore requires labeled training data. However, labeled training data is scarce, especially for languages other than English. One approach to address this issue is to enlarge training data in a semi-supervised fashion (Severyn and Moschitti, 2015). Instead, we propose to make use of already available linguistically motivated resources. Especially sentiment lexicons are important cues for polarity classification (cf. Mohammad et al., (2013)).

Our contributions in this chapter are:

1. We introduce two intuitive and simple methods of incorporating linguistic features into a CNN. The resulting architecture is called *linguistically-informed Convolutional Neural Network (lingCNN)*. The first method is to add features to every word in a sentence. That enables the model to learn interactions between words and between individual word embeddings and linguistic features. For example the Twitter text "[...] it's not like there was a viable 2nd option [...]" contains the usually positive word "like", which would be negated by "not". A CNN is capable of learning that the sense of "like" is different here. That plus word-level linguistic features allows the lingCNN to learn phrases and their polarity.

2. The second method is to add feature vectors that are computed based on the entire sentence.

3. The results show that word-level features can improve the classification and are more beneficial than sentence-level features. However, the combination of both methods reaches the best performance, indicating that both feature types are to some extend orthogonal. Our best results are comparable to state-of-the-art on the SemEval Twitter polarity data set.

4. In our analysis we show that linguistic features are especially beneficial if there is little training data. This fact makes lingCNN especially suitable for under-resourced languages.

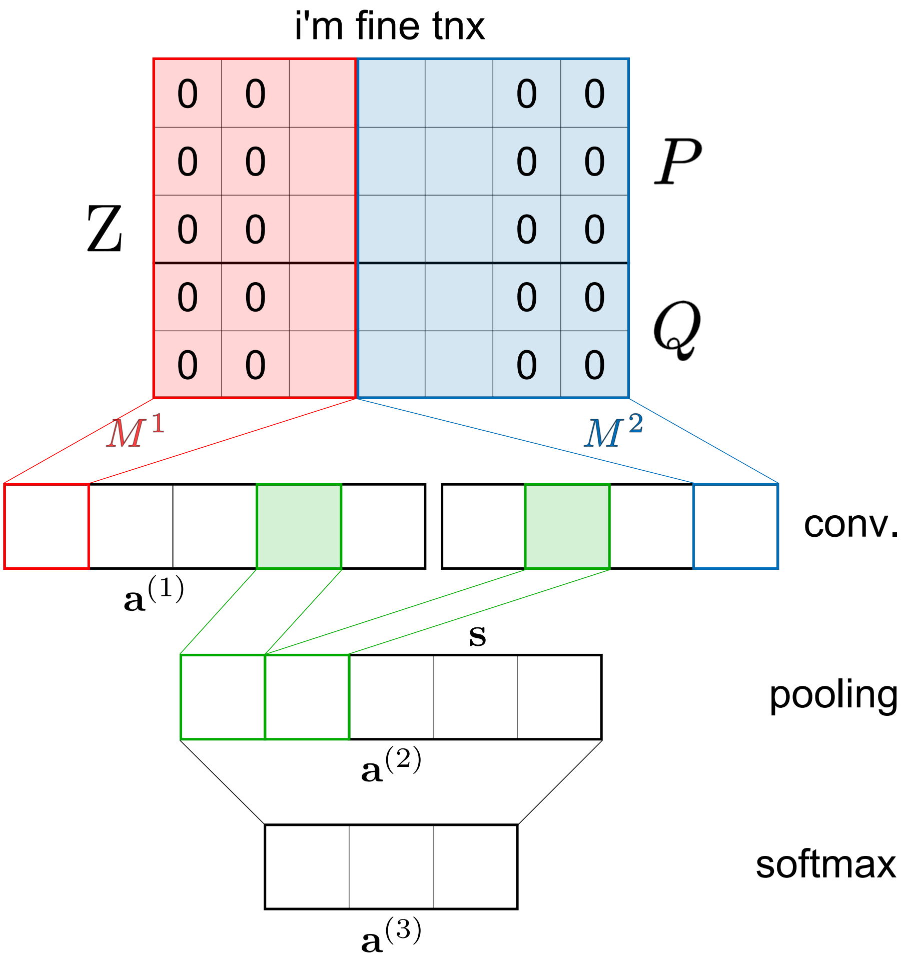


FIGURE 5.2.1: LINGCNN ARCHITECTURE LINGCNN architecture with word- and sentence-level features.

Section 5.2 introduces our extensions to the standard CNN architecture. In Section 5.2.1 and Section 5.2.2 we present word-level and sentence-level features. The experiments are described in Section 5.3. The results are followed by the analysis (Section 5.4) and related work (Section 5.5). Finally, the chapter is concluded in Section 5.6.

5.2 LINGCNN ARCHITECTURE

After introducing the general CNN foundations in Chapter 2, we now present adaptations for incorporating linguistic knowledge.

Figure 5.2.1 depicts the LINGCNN architecture.

5.2.1 *Word-level Features*

In Chapter 2, Equation 2.2.9 we have defined the input of the CNN as:

$$Z = \begin{bmatrix} | & | & | \\ LT_{\cdot,t_1} & \cdots & LT_{\cdot,t_n} \\ | & | & | \end{bmatrix}$$

where each word t_i is represented by a d dimensional vector in the lookup table LT. To incorporate linguistic features at word-level into the learning process we create the lookup table by concatenating two matrices: $LT = \begin{bmatrix} P \\ Q \end{bmatrix}$. $P \in \mathbb{R}^{d_P \times |V|}$ denotes a matrix of low-dimensional word embeddings, learned for example with a Neural Network Language Model (NNLM), such as the Log-Bilinear Language model (LBL). d_P, the size of the embeddings, is usually set to $50 - 300$, depending on the task.

In addition to P, we introduce another matrix $Q \in \mathbb{R}^{d_Q \times |V|}$, which contains external word features. In this case d_Q is the number of features for a word. The features in Q are precomputed and not embedded into any embeddings space, i.e., Q is fixed during training. We use the following feature types:

BINARY SENTIMENT INDICATORS These features indicate a word's *prior* polarity as given by lexicons. We create two such features per word per lexicon. The first feature indicates positive and the second negative polarity of that word in the lexicon. Having two separate features allows us to indicate if a word can be both positive and negative.

The lexicons used for this feature type are the Opinion lexicon (Hu and Liu, 2004), MPQA (Wilson et al., 2005), and NRCC Emotion lexicon (Mohammad and Turney, 2013).

SENTIMENT SCORES The Sentiment140 lexicon and the Hashtag lexicon (Mohammad et al., 2013). Both lexicons have been explicitly developed for the Twitter domain. They provide a valence score for each word instead of just a binary label. A positive score indicates positivity, a negative score negativity. The higher the absolute number of the score, the stronger the sentiment conveyed by a word is. We directly incorporate these scores into the feature matrix. Please note that we do not need a separate feature for positive and negative here (in contrast to binary sentiment indicators), because the two lexicons do not provide several scores for one word.

SENTIMENT NGRAM SCORES Both lexicons, Sentiment140 lexicon and the Hashtag lexicon, also contain scores for bigrams. For example the bigram "lazy saturday" is labeled as positive (it has a score of 5). Both lexicons furthermore contain skip ngrams, which are a sequence of a uni- or bigram, followed by a sequence

feature type	example	value
binary positive	cute	positive
binary negative	annoying	negative
score unigram positive	cute	0.1
score unigram negative	annoying, find	-0.9, -0.1
score bigram positive	very cute	1.6
score bigram negative	so annoying	-1.5
score skip ngram positive	i * sun	1.3
score skip ngram negative	i * so annoying	-5
emoticon positive	:)	1
emoticon negative	:(	-1
negation words	don't, never	-
punctuation	,.!?	-
POS	A(djective), E(moticon), V(erb), O(ther)	-

table 5.2.1: example of linguistic resources Exemplary items of several linguistic resources as described in the word level features. The sentiment scores are taken from the Hashtag Lexicon (Mohammad et al., 2013).

of arbitrary words, followed by another uni- or bigram. For instance, the skip ngram "i * so annoying" (labeled as negative with a score of -5) would match "i don't find him so annoying" or "i think it's so annoying". In both cases, bigram or skip ngrams, all words of the sequence receive the same score that is assigned by the lexicon.

binary negation Following Christopher Potts,[1] we mark each word between a negation word, such as "never" or "not" and the next punctuation, such as a period or a comma, as negated.

In total each word receives 13 additional features ($3 * 2$ binary, 2 unigram scores, $2 * 2$ (skip) ngram scores, 1 negation). Since lingCNN performs a 2d convolution over all feature dimensions, it allows the detection of features that interact with word embeddings and linguistic features.

Lets consider the example sentence: "i don't find him so annoying , but cute :)". For the example resources listed in Table 5.2.1 the features are shown in Table 5.2.2.

[1] http://sentiment.christopherpotts.net/lingstruc.html

	i	don't	find	him	so	annoying	,	but	cute	:)
Part-of-Speech (POS)	O	O	V	O	O	A	O	O	A	E
binary positive	0	0	0	0	0	0	0	0	1	0
binary negative	0	0	0	0	0	1	0	0	0	0
score unigram	0	0	-0.1	0	0	-0.9	0	0	0.1	0
score bigram	0	0	0	0	-1.5	-1.5	0	0	0	0
score skip ngram	-5	-5	-5	-5	-5	-5	0	0	0	0
binary negation	0	0	1	1	1	1	0	0	0	0

table 5.2.2: word-level feature matrix for example sentence Linguistic features for the example sentence "i don't find him so annoying , but cute :)".

5.2.2 *Sentence-level Features*

An alternative to adding word-level features into the training process is to add sentence-level features. The reason for doing so is that simple count features work surprisingly well (e.g., Mohammad et al., (2013)). In lingCNN these features are concatenated with the pooling layer's output to serve as additional input for the softmax layer. We recall the pooling output in Equation 2.2.4 being defined as:

$$\mathbf{a}^{(2)} = \max(0, \mathbf{a}^{(1)} + b^{(2)})$$

We simply redefine $\mathbf{a}^{(2)}$ as the concatenation of the activated values and the sentence-level feature vector $\mathbf{s}$:

$$\mathbf{a}^{(2)} = [\max(0, \mathbf{a}^{(1)} + b^{(2)})\ \mathbf{s}] \tag{5.2.1}$$

The definition for k-max pooling (Equation 2.2.10) is accordingly.

counts We count the number of elongated words such as "coooool", because they frequently express sentiment. A word is considered elongated when it contains at least three equal characters in a row.

Another feature is the count of emoticons, where the list of possible emoticons is taken from the SentiStrength project.[2] Further, we count the number of contiguous sequences of punctuation, such as "..." or "!!!". And finally we count the number of negated words using the same list of words as in the word-level features.

[2] http://sentistrength.wlv.ac.uk/

feature type	value			
no. of elongated words	0			
no. of emoticons	1			
no. of punctuation sequences	0			
no. of negated words	4			
scores Tweet	3	-0.9	0.1	0.1
scores hashtag	0	0	0	0
scores adjective	2	-0.8	0.1	0.1
scores emoticon	1	1	1	1
scores verb	1	-0.1	0	-0.1
scores other POS	0	0	0	0

Table 5.2.3: Sentence-level feature matrix for example sentence Linguistic features for the example sentence "i don't find him so annoying , but cute :)". The scores are only shown exemplary for unigrams (*score unigram* in Table 5.2.1) and for emoticons.

Sentiment scores Mohammad et al., (2013) showed that simple sentence-level sentiment features can be very successful. Therefore, we reimplement their feature set. The computed lexicon features are the number of sentiment words in a sentence, the sum of sentiment scores of these words as provided by the lexicons, the maximum sentiment score, and the sentiment score of the last analyzed word. These four numbers are calculated for all 5 previously mentioned sentiment lexicons: Opinion lexicon (one time) (Hu and Liu, 2004), MPQA (one time) (Wilson et al., 2005), NRCC Emotion lexicon (one time), Sentiment140 lexicon (three times for uni-, bigrams, and skip ngrams), and the Hashtag lexicon (three times) (Mohammad et al., 2013). Moreover, these features are computed separately for the entire sentence, for each POS tag (25 as described below) and for all hashtag tokens in the sentence (Mohammad et al., 2013).

The total number of sentence-level features is 976 (4 count features, $(25+2)*4*9$ sentiment scores).

In order to be able to compute the feature vector for the previously seen example "i don't find him so annoying , but cute :)" it needs to be tagged with POSs. Using the example resources in Table 5.2.1 the POS sequence is "O O V O O A O O A E" (cf. Table 5.2.2). The resulting sentence-level features are listed in Table 5.2.3.

5.3 EXPERIMENTS

5.3.1 *Data*

To evaluate LINGCNN, we use the SemEval 2015 Task 10B data set (Rosenthal et al., 2015). SemEval is a collection of shared tasks each dealing with a different topic in semantics. The task we use is Task 10: *Sentiment Analysis in Twitter*, Subtask B *Message polarity classification* Rosenthal et al., (2015). Here the task is to classify entire text messages (Twitter tweets and SMS) into positive, negative, and neutral.

Equally to the official shared task we train the model on the SemEval 2013 training and development set and use the SemEval 2013 test set as development set (Nakov et al., 2013; Rosenthal et al., 2015). This leads to 9845 tweets in the training set and 3813 tweets in the development set. The final evaluation is done on the SemEval 2015 test set, which contains 2390 tweets. Table 5.3.1 lists all data set sizes and the label distribution in detail. We can see that the negative class is strongly underrepresented.

Additionally, to compare with other SemEval 2015 participants, we use the SMS dataset from SemEval 2013 (Nakov et al., 2013), and the Twitter, Twitter sarcasm, and LiveJournal datasets from SemEval 2014 (Rosenthal et al., 2014). Moreover, we test the generality of our findings by reporting results on the manually labeled test set of the Sentiment140 corpus (Go et al., 2009). It contains about 500 tweets (cf. Table 5.3.1), which were collected by searching Twitter for specific categories, such as movie, person, and company. Table 5.3.1 shows the details of all datasets.

The examples in all data sets are labeled with one of the three classes: *positive*, *negative*, or *neutral*. As proposed by the SemEval organizers, tweets labeled as *objective* are mapped to the *neutral* label. We report accuracy and the macro F_1 score of the positive and negative classes, because this is the official shared task evaluation metric:

$$F_{1,\text{macro}} = \frac{\left(F_{1,\text{positive}} + F_{1,\text{negative}}\right)}{2} \tag{5.3.1}$$

Data Prepocessing

The SemEval and Sentiment140 data are preprocessed in the following way:

TOKENIZATION AND POS TAGGING Tweets are first tokenized and POS tagged using TweetNLP (Owoputi et al., 2013). It has been developed especially for Twitter and therefore can handle frequent phenomena that a standard tokenizer/tagger cannot handle very well. Examples are correct tokenization of emoticons and correct POS tagging of interjections (e.g., "lololol", "ikr" - "I know right") and proper nouns (e.g., "fb" - "Facebook").

	TOTAL	POSITIVE	NEGATIVE	NEUTRAL
Twitter 2015 train	9845	3636	1535	4674
Twitter 2015 dev	3813	1572	601	1640
Twitter 2015 test	2390	1038	365	987
Sentiment140 test	498	182	177	139
SMS 2013	2093	492	394	1207
Twitter 2014	1853	982	202	669
Twitter 2014 sarcasm	86	33	40	13
LiveJournal 2014	1142	427	304	411

TABLE 5.3.1: TWITTER DATASET SIZES Number of overall, positive, negative, and neutral Twitter tweets/SMS per dataset.

NORMALIZATION In the Twitter domain there are certain standards that are unique. For example, words starting with "@" refer to other users, and words starting with "#" (so called *hashtags*) describe thoughts or feelings. Additionally, tweets very often contain web URLs. Neither URLs, nor user mentions do provide any cue of polarity. Therefore, we normalize them to "<web>" and "<user>". We keep hashtags, because they often contain valuable information such as topics or even sentiment (e.g., "#happyday").

Punctuation sequences like "!?!?" can act as exaggeration or other polarity modifiers, thus we want to keep them. However, the sheer amount of possible sequences increases the Out-of-Vocabulary (OOV) rate dramatically. Therefore, we normalize them in the following way. All sequences of punctuations are replaced by a list of distinct punctuations in this sequence (e.g., "!?!?" is replaced by "[!?]"). Additionally, we sort the remaining characters to lower the variability even further. This way, we keep most of the information without increasing the vocabulary size or the OOV rate much. We consider the following punctuation characters: ".:;!?,!?\'\-_<>*".

LOWERCASING AND SHUFFLING In the next step, we lowercase all tweets to further reduce the vocabulary size. This is an important step especially for Twitter, because of the great variety of (mis-)spellings of words. Finally, the datasets are randomly shuffled.

5.3.2 *Model Settings*

Baseline Systems

The first baseline is the majority baseline, i.e., a classifier that would always predict the most common class. Since we use the macro F_1 of the positive and negative class, we have to choose the most frequent class among these two. In both Twitter 2015 and Sentiment140 that is the positive class.

We use the SemEval 2013 and SemEval 2014 winning system (Mohammad et al., 2013) as baseline. This system uses a Support Vector Machine (SVM) for classification. According to their analysis, Bag-of-Words (BOW) features and linguistic features are the most important ones. BOW features are computed for words ($\{1, 2, 3\}$-grams) and for characters $\{3, 4, 5\}$-grams. Linguistic features are the ones we use as sentence-level features for lingCNN: counts and sentiment scores (cf. Section 5.2.2). Features such as POS tags or clusters have not made an important contribution in the experiments of Mohammad et al., (2013). Therefore, we implement only BOW and linguistic features. To account for differences in scales of the different sentiment scores, we standardize them to have a zero mean and a standard deviation of 1. In preliminary experiments this slightly improved the results.

There are three feature settings we analyze: (i) only BOW features (for both, word and characters), (ii) only linguistic features, and (iii) the combination of BOW and linguistic features. We use LIBLINEAR (Fan et al., 2008) to train the model and optimize the C parameter on the development set. The analyzed values are $C \in \{1e-4, 5e-4, 1e-3, 5e-3, 1e-2, 5e-2, 1e-1, 5e-1, 1, 2, 3, 5, 7\}$.

For reference we add the first and second best systems of the SemEval 2015 tweet level polarity task: Webis (Hagen et al., 2015) and UNITN (Severyn and Moschitti, 2015). Webis is an ensemble based on four systems, which participated in the same task of SemEval 2014 (Task 9, subtask B). One of it is our SVM baseline, the others are (i) a Stochastic Gradient Descent (SGD) classifier with mainly linguistic features (Günther and Furrer, 2013), (ii) Maximum Entropy classifier with statistical and linguistic features (Proisl et al., 2013), and (iii) a system that is similar to our SVM baseline with more POS, word- and n-gram features, and more sentiment lexicons (Miura et al., 2014). The UNITN system trains a CNN similar to ours. They rely on pretraining the entire model on a large distant supervised training corpus (10M labeled tweets). This approach is orthogonal to ours and can easily be combined with our idea of linguistic feature integration. This combination is likely to increase the performance further.

LingCNN

To analyze the effect of the linguistic features and our extensions we train different CNN models with different combinations of features: (i) only pretrained word embeddings, (ii) integration of word-level features, and (iii) integration of sentence-level features. The model updates all parameters during training $\theta = \{P, M^*, W, b^{(*)}\}$, where P is the embeddings matrix, M^* are the filter matrices, W is the weight matrix of the softmax layer, and b^* are the model's biases (see Section 2.2). We set the embeddings size to $d_P = 60$. Our model uses filters of width $2 \leq m \leq 5$ with 100 filters each and set k-max pooling to $k = 1$. We train the models for a maximum of 30 epochs with mini-batch SGD (batch size: 100). The training was stopped when three consecutive epochs lead to worse results on the development set (early stopping). We use AdaGrad (Duchi et al., 2011) for dynamic learning rate adjustment with an initial learning rate of $\eta = 0.01$ and ℓ_2 regularization ($\lambda = 5e^{-5}$).

The embeddings matrix P is initialized in two different ways. First, we pretrain Twitter specific word embeddings, because previous work has shown that pretrained word embeddings are helpful in various tasks (e.g., Kim, (2014)). In order to do so we train skip-gram word embeddings (Mikolov, Chen, et al., 2013) with the word2vec toolkit[3] on a large amount of unlabeled Twitter text data. We first downloaded about 60 million tweets from the unlabeled Twitter Events data set (McMinn et al., 2013). It is preprocessed the same way as the other datasets. The vocabulary is built out of all the words of the SemEval training data and the 50K most frequent words of the Twitter Events data set. This way we increase the chance to have good embeddings for frequent words in the SemEval test set. Additionally, an *unknown* word is added to the vocabulary to learn a word embedding for out-of-vocabulary words. Every word that does not exist in this vocabulary is replaced by the unknown word. Finally, a skip-gram model with 60-dimensional vectors is trained on the unlabeled data and used to initialize the word embeddings matrix P. The matrix P is as stated above further fine-tuned during model training.

As second word embeddings initialization method we create random 60 dimensional embeddings for all words in the same vocabulary. Each random number is sampled from a normal distribution with a mean of 0 and a standard deviation of 0.01.

5.3.3 *Results*

Baselines

Table 5.3.2 lists the baseline results on the SemEval 2015 and the Sentiment140 test sets. As expected, the majority baseline yields the lowest

[3] https://code.google.com/p/word2vec/

	SemEval 2015		Sentiment140	
model	acc.	F_1	acc.	F_1
majority (positive)	43.43	30.28	36.55	26.76
SVM BOW ($C = 0.005$)	62.13	51.07	68.67	67.94
SVM ling. ($C = 0.0001$)	64.90	57.88	66.67	66.61
SVM BOW + ling. ($C = 0.001$)	66.53	59.32	70.21	70.08
Webis	-	64.84	-	-
UNITN	-	64.59	-	-

table 5.3.2: baseline results Test set results of the baseline systems. SVM is our reimplemented SVM with a BOW and linguistic features similar to Mohammad et al., (2013). C is the value of the C parameter of the SVM that yielded the best result on the development set.
Webis (Hagen et al., 2015) is an ensemble system of 4 individual classifiers. UNITN (Severyn and Moschitti, 2015) is a CNN trained on a large distant supervised corpus. Webis and UNITN only provide F_1 results on the SemEval test set.

performance. Especially on Sentiment140, where the number of positive and negative examples are almost equal and there are almost as many neutral examples, this baseline is very weak. The other baselines are much stronger.

Similar to Mohammad et al., (2013)'s findings, the combination of ngram and linguistic features gives the best performance for the acsvm. We can also see that linguistic features alone are more valuable than just ngram features. This shows how important linguistic resources are for this classification task. The fact that the SVM with bow features is only 2.8% in accuracy behind the SVM with linguistic features, but almost 7 F_1 points indicates that the former has big trouble classifying the negative class. Our analysis proves that.

Interestingly, the ngram SVM is better than the linguistic only SVM on the Sentiment140 dataset. In general, the results on this test set are better by a large margin, indicating that the SemEval data set is more difficult and more different than the training data.

Both SemEval participating systems beat even the best SVM baseline by a large margin.

LingCNN

Table 5.3.3 shows the lingCNN results on the SemEval 2015 test set. With only word-level features the model yields similar F_1 performance as the SVM with only linguistic features. Adding sentence-level features improves the performance to the level of the SVM baseline sys-

	features				SemEval 2015		Sentiment140	
	w2v	rand	word	sentence	acc.	F_1	acc.	F_1
1			+		61.80	57.83	69.08	72.58
2			+	+	63.51	59.24	71.49	74.36
3		+			64.90	58.50	70.28	70.40
4		+		+	65.27	58.89	74.90	76.73
5		+	+		66.40	62.22	76.51	78.67
6		+	+	+	66.23	62.10	76.91	79.10
7	+				67.36	62.72	76.31	77.59
8	+			+	66.95	62.61	77.71	79.14
9	+		+		67.41	63.43	**78.71**	80.21
10	+		+	+	**68.16**	**64.46**	78.31	**80.75**

table 5.3.3: lingCNN results Test set results of lingCNN for different feature type combinations. "w2v" are pretrained word embeddings; "rand" are randomly initialized embeddings; "word" are word-level features; "sentence" are sentence-level features. A "+" indicates that the feature type is active.

tem with bag-of-words and linguistic features. This was the winning system of the SemEval 2014.

Random embeddings as only feature yields lower F_1 performance than the combination of word- and sentence-level features. Similarly, adding sentence-level features is worse. When adding word-level features, the performance increases by a large range, e.g., 3 F_1 points from configuration 2 to configuration 5.

We see that using pretrained word embeddings as only feature type yields large improvements. Sentence features on top of that can not improve the performance further. However, word-level features together with pretrained word embeddings yield higher performance. The best result on the SemEval 2015 test set is reached by the combination of word embeddings and both types of linguistic features. This performance is comparable with both state-of-the-art SemEval 2015 winner systems (cf. Table 5.3.2).

A general finding is that linguistic features seem very beneficial if there is little domain-specific knowledge. Pretrained embeddings, which are trained on domain-specific data, cannot benefit as much from them as random embeddings can: 1.74 F_1 from configuration 7 to configuration 10, compared to 3.6 from configuration 3 to 6.

Significance – again computed using the approximate randomization test (Padó, 2006) – of all results from Table 5.3.3 is shown in Table 5.3.4. We can confirm that word embeddings are necessary to reach high performance. However, the right initialization of embeddings is important. Configuration 6 in Table 5.3.3 (random embeddings plus word-level plus sentence-level features) is significantly worse than its pretrained counterpart in configuration 10. That is consistent with findings of previous work on CNNs (e.g., Kim, (2014)). Additionally, word-level features are confirmed to reach better performance than sentence-level features, because the difference between configuration 8 and configuration 10 is significant, whereas the difference between configuration 9 and 10 is not.

The results on the Sentiment140 test set (right part of Table 5.3.3) show the same tendencies. Linguistic features help the classifier to a great extend. This time even the combination of pretrained word embeddings and sentence-level features yields better results than just the word embeddings alone. The best F_1 score again is reached by the model with pretrained embeddings and both types of linguistic features.

Table 5.3.5 shows the official SemEval 2015 results. The rank of a system for every dataset is shown in subscript. The datasets are named according to the organizer's convention. *Twitter 2013* is the SemEval 2013 test set, which is used as SemEval 2015 development set. *Twitter 2015* is the SemEval 2015 test set.

Compared to the 30 best systems of SemEval 2015 LINGCNN would rank 3rd (last column in Table 5.3.5). Our system further scores well on *LiveJournal* (2nd) and *SMS* (4th). These two datasets are probably less noisy than the other datasets that are Twitter-based. Since *Twitter 2013* is the SemEval 2015 development set, one can see that the development data (Twitter 2013) and test data (Twitter 2015) are quite different, because all systems loose performance.

Please also note that *CIS-positive* is our official SemEval 2015 submission system (Ebert et al., 2015b), which is a more basic version of LINGCNN.

5.4 ANALYSIS

5.4.1 *Examples*

Here, we analyze examples to find out why the linguistic features help. Consider the example "saturday night in with toast , hot choc & <user> on e news #happydays". Only the hashtag "#happydays" indicates polarity. The hashtag exists in the hashtag sentiment lexicon (Mohammad et al., 2013), but does not exist in the training vocabulary. Therefore, there is no embedding for it, rather the *unknown* word embedding is used. Thus, a standard CNN does not have any information about contexts or sentiment of this token.

Here is another example: "shiiiiit my sats is on saturday . i'm going to fail". "Fail" is strongly negative in all lexicons. However, it occurs only 10 times in the training set. That is likely not enough to learn

	1	2	3	4	5	6	7	8	9	10
1										
2	†									
3										
4										
5										
6	‡	‡								
7	‡	‡								
8	‡	‡								
9	‡	‡								
10	‡	‡				‡		∗		

TABLE 5.3.4: SIGNIFIANCE Significant differences of lines 1–10 in Table 5.3.3. ‡: $p = 0.01$, †: $p = 0.05$, ∗: $p = 0.1$.

a good sentiment-bearing embedding. As a result, the CNN without linguistic knowledge classifies the tweet as neutral. Having linguistic features enables the model to implicitly incorporate sentiment information into the word embeddings, helping to classify this example correctly. Note that proper normalization of "shiiiiit" might have given the model another clue of negative polarity.

5.4.2 *Corpus Size*

In this section we analyze the benefit of linguistic features with respect to the size of the training corpus. Figure 5.4.1 and Table 5.4.1 shows the performance of a standard CNN with word embeddings and LINGCNN with both types of linguistic features. The two models are trained on different fractions of the SemEval training set. We clearly see that linguistic features are helpful in all cases. Especially, where only limited training data is available, the performance difference is large. Even with only 1000 training samples, LINGCNN yields a reasonable result of 60.89. The CNN that does not have access to linguistic features reaches only 49.89. Although, the performance of the standard CNN without linguistic features increases much for 3000 training examples, this model is still more than 4 points behind the linguistically informed model. The more training data is available the smaller is the difference between both models. But still, even when using the entire training set, LINGCNN yields higher performance.

	Twitter	SMS	Twitter	Sarcasm	LiveJournal	Twitter
	2013	2013	2014	2014	2014	2015
Webis	68.49_{11}	63.92_{15}	70.86_{8}	49.33_{11}	71.64_{14}	$\mathbf{64.84_{1}}$
unitn	72.79_{2}	68.37_{2}	73.60_{2}	55.44_{5}	72.48_{13}	64.59_{2}
lingCNN	69.71_{8}	67.76_{4}	70.90_{6}	46.67_{18}	74.69_{2}	64.46_{3}
lsislif	71.34_{4}	63.42_{18}	71.54_{5}	46.57_{19}	73.01_{11}	64.27_{4}
INESC	71.97_{3}	63.78_{16}	72.52_{3}	56.23_{3}	69.78_{22}	64.17_{5}
Splusplus	$\mathbf{72.80_{1}}$	67.16_{6}	$\mathbf{74.42_{1}}$	42.86_{26}	$\mathbf{75.34_{1}}$	63.73_{6}
wxiaoac	66.43_{17}	64.04_{14}	68.96_{12}	54.38_{7}	73.36_{10}	63.00_{7}
IOA	71.32_{5}	68.14_{3}	71.86_{4}	51.48_{9}	74.52_{3}	62.62_{8}
Swiss-Chocolate	68.80_{10}	65.56_{7}	68.74_{13}	48.22_{14}	73.95_{5}	62.61_{9}
CLaC-SentiPipe	70.42_{7}	63.05_{19}	70.16_{11}	51.43_{10}	73.59_{7}	62.00_{10}
TwitterHawk	68.44_{12}	62.12_{21}	70.64_{10}	56.02_{4}	70.17_{19}	61.99_{11}
SWATCS65	68.21_{13}	65.49_{9}	67.23_{15}	37.23_{31}	73.37_{9}	61.89_{12}
UNIBA	61.66_{28}	65.50_{8}	65.11_{26}	37.30_{30}	70.05_{20}	61.55_{13}
KLUEless	70.64_{6}	67.66_{5}	70.89_{7}	45.36_{23}	73.50_{8}	61.20_{14}
NLP	66.96_{15}	61.05_{25}	67.45_{14}	39.87_{27}	66.12_{30}	60.93_{15}
ZWJYYC	69.56_{9}	64.72_{12}	70.77_{9}	46.34_{20}	71.60_{15}	60.77_{16}
Gradiant-Analytics	65.29_{21}	61.97_{22}	66.87_{18}	$\mathbf{59.11_{1}}$	72.63_{12}	60.62_{17}
IIIT-H	65.68_{19}	62.25_{20}	67.04_{17}	57.50_{2}	69.91_{21}	59.83_{18}
ECNU	65.25_{22}	$\mathbf{68.49_{1}}$	66.37_{21}	45.87_{22}	74.40_{4}	59.72_{19}
CIS-positiv	64.82_{23}	65.14_{11}	66.05_{22}	49.23_{12}	71.47_{16}	59.57_{20}
SWASH	63.07_{26}	56.49_{31}	62.93_{30}	48.42_{13}	69.43_{24}	59.26_{21}
GTI	64.03_{24}	63.50_{17}	65.65_{23}	55.38_{6}	70.50_{17}	58.95_{22}
iitpsemeval	60.78_{30}	60.56_{26}	65.09_{27}	47.32_{16}	73.70_{6}	58.80_{23}
elirf	57.05_{31}	60.20_{28}	61.17_{31}	45.98_{21}	68.33_{28}	58.58_{24}
SWATAC	65.86_{18}	61.30_{24}	66.64_{20}	39.45_{28}	68.67_{27}	58.43_{25}
UIR-PKU	67.41_{14}	64.67_{13}	67.18_{16}	52.58_{8}	70.44_{18}	57.65_{26}
SWATCMW	65.67_{20}	65.43_{10}	65.62_{24}	37.48_{29}	69.52_{23}	57.60_{27}
WarwickDCS	66.57_{16}	61.92_{23}	65.47_{25}	45.03_{25}	68.98_{25}	57.32_{28}
SenticNTU	63.50_{25}	60.53_{27}	66.85_{19}	45.18_{24}	68.70_{26}	57.06_{29}
DIEGOLab	62.49_{27}	58.60_{30}	63.99_{28}	47.62_{15}	63.74_{31}	56.72_{30}

table 5.3.5: semeval 2015 results Best 30 systems from the official results of the SemEval 2015 shared task on several test sets. lingCNN is highlighted in bold. Numbers in subscript indicate the rank according to the corresponding test set. The best result per column is highlighted in bold.

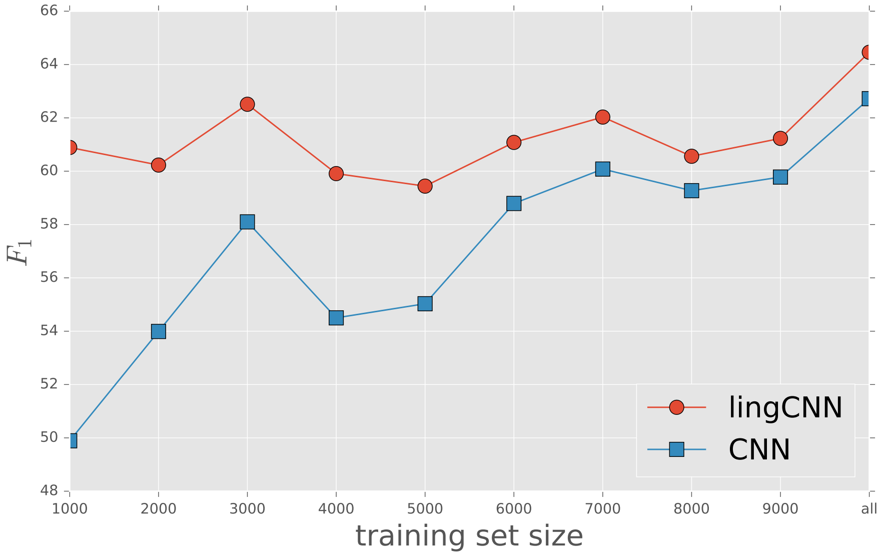


FIGURE 5.4.1: ANALYSIS OF TRAINING SET SIZES Comparison of a standard CNN with lingCNN for different training set sizes.

SIZE	CNN	lingCNN
1000	49.89	60.89
2000	53.99	60.23
3000	58.10	62.51
4000	54.50	59.91
5000	55.03	59.44
6000	58.79	61.08
7000	60.08	62.03
8000	59.27	60.56
9000	59.78	61.23
all	62.72	64.46

TABLE 5.4.1: ANALYSIS OF TRAINING SET SIZES Comparison of a standard CNN with lingCNN on different training set sizes.

5.5 RELATED WORK

Collobert et al., (2011) published the first CNN architecture for a range of natural language processing tasks, such as chunking and Named Entity Recognition (NER). They propose to use multiple look-up tables and add simple features indicating if a word exists in a gazetteer list. We adopt their idea of using multiple look-up tables to incorporate linguistic features at the word-level into the CNN and add more feature type variants.

Since then CNNs have been used for a variety of sentence classification tasks (e.g., Zeng et al., (2014)), including polarity classification (e.g., Kim, (2014)). Kalchbrenner et al., (2014) showed that their DCNN for modeling sentences can achieve competitive results in this field. They introduce several techniques that increase model complexity. The techniques are (i) wide convolution; (ii) *k*-max pooling, that uses the *k* maximum values instead of only single largest number after convolution; (iii) dynamic *k*-max pooling that chooses *k* according to the input sentence length; (iv) folding, a special subsampling technique that reduces the number of parameters. Our CNN architecture is simpler than theirs. We use max pooling and a single layer only. Having more layers would lead to overfitting on the small SemEval dataset. We also use wide convolution but use multiple filter widths in the convolution layer.

There are alternative approaches of integrating linguistic features into model training. By adding more labeled data, implicit knowledge is given to the model. This approach usually requires manual labeling effort. Alternatively, a method called *distant supervision* semi-automatically labels texts based on emoticons it contains (Go et al., 2009). This approach is used by Severyn and Moschitti, (2015) to train a CNN. Please note that adding more training data is orthogonal to our approach.

A different method to integrating linguistic features is to incorporate linguistic knowledge into the objective function to guide the model training. For instance Tang, Wei, Yang, et al., (2014) incorporate the polarity of an ngram into a hinge loss function.

Tang, Wei, Qin, Liu, et al., (2014) used a CNN to compute representations of input sentences. These representations together with linguistic features on sentence-level form the input to an SVM. In contrast, we use linguistic features at the word-level, which allows interaction between linguistic features and word embeddings. Furthermore, we use similar sentence features and directly incorporate them into the CNN.

In addition to CNNs, researchers have been using different neural network architectures. However, each of these has its own disadvantages. A deep feed forward network cannot model easily that inserting many types of words into a string (e.g., "happy to drive my new car" vs "happy to drive my *red* new car") does not change sentiment. acprnn (Elman, 1990) and Long Short Term Memorys (LSTMs) (Hochreiter and Schmidhuber, 1997) are powerful for unbounded dependencies, but tweets are short; the sentiment of a tweet is usually determined by one part of it and unlike Recurrent Neural Network (RNN)/LSTM, convolution plus max pooling can learn to focus on that. Recursive architectures like the Recursive Neural Tensor Network (Socher et al., 2013) assume some kind of hierarchical sentence structure. This structure does not exist or is hard to recognize for many noisy tweets.

As mentioned before, we use the SemEval 2013 and SemEval 2014 winning system (Mohammad et al., 2013) as baseline. Moreover, we include several features of their system to improve the CNN.

5.6 conclusion

In this chapter we have shown that CNNs are powerful classifiers for the task of sentence-level polarity classification. We have introduced an intuitive and simple way of incorporating linguistic word-level and sentence-level features into a standard CNN architecture. Using such features yields significant improvements on two polarity classification Twitter data sets without the need of more labeled training data. Using both feature types, our lingCNN performs comparable to state-of-the-art systems of the SemEval 2015 shared task.

Our analysis shows that especially (but not only) when little domain knowledge in form of training data or pretrained word embeddings is available lingCNN is more effective than an uninformed standard CNN. This suggests that it especially helps when the model is not powerful enough to capture all the data's variance.

Our proposed methods are easy to implement and often times do not require new resources, because there often are already some available depending on the task.

5.7 future work

The following points are possible extensions to the existing system:

- More often than not fine-tuning word embeddings during the training on task-dependent data improves the performance of models (e.g., Kim, (2014)). Along these lines linguistic word-level features could be fine-tuned during training as well. Starting with a lexicon's *prior* polarity score (or label transformed into a score) domain specific scores can be the result. For instance "read" can be a positive word in the book domain, whereas it can be negative in the movie domain where "go read the book" is a negative statement.

- More linguistic features can enhance the system even further. For instance features indicating uncertainty (“might”, “may”) change polarity. POS-based features can help the model to disambiguate between different uses and are already given by the MPQA lexicon (Wilson et al., 2005). Mohammad et al., (2013) have been successful with ngram features; we could incorporate them as additional sentence-level feature type. Both word and character ngram features are possible. More Twitter-specific features can be extracted from elongated words or hashtags.

6 MORPHOLOGICALLY INDEPENDENT SENTIMENT ANALYSIS

> This chapter covers work already published at international peer-reviewed conferences. The relevant publication is Ebert et al., (2016). The research described in this chapter was carried out in its entirety by myself. The other author(s) of the publication(s) acted as advisor(s) or were responsible for work that was reported in the publication(s), but is not included in this chapter.

In this chapter we want to find out what the role of morphology is on polarity classification. The underlying hypothesis is that morphology is not relevant for polarity classification. Whereas for valence prediction it is important to know that *better* may have a more positive connotation than *good*, for polarity it may not make a difference.

In this chapter we address this question by neglecting all derivational and inflectional morphology by learning stem- and lemma-based word embeddings that can be used in classifiers such as linguistically-informed Convolutional Neural Network (lingCNN). Neglecting morphology enables us to map all forms of sentiment-bearing words, such as *love* and *hate,* to one canonical form each. This makes a system more robust against rare or unseen word forms. Additionally, stem- and lemma-based word embeddings have the benefit of working especially well for morphologically rich languages.

Instead of training embeddings of surface forms of words, we train embeddings of lemmata. This yields significantly better results than standard word embeddings in three experiments. On a new WordNet-based evaluation STem EMbeddings (Stem) and LemmA eMBeddings (Lamb) lead to significant improvements for five different languages. They are up to 50% better than standard embeddings. A strong improvement is also shown on popular word similarity and polarity classification tasks. Additionally, our analysis shows that lemma embeddings successfully address sparsity and therefore lead to more efficiency because high-quality embeddings can be learned even for smaller dimensionalities and for smaller training corpora.

6.1 INTRODUCTION

Despite their power and prevalence, embeddings have serious practical problems. First, large text corpora are necessary to train high-quality embeddings. Such corpora are not available for under-resourced languages. Second, Morphologically Rich Languages

GENUS / NUMERUS	FORM	STEM
infinitive	brechen	brech
1st singular	breche	brech
2nd singular	brichst	brich
3rd singular	bricht	bricht
1st plural	brechen	brech
2nd plural	brecht	brecht
3rd plural	brechen	brech

TABLE 6.1.1: STEMMING RESULT OF "BRECHEN" Result of stemming of the German verb "brechen" (to break) for all present indicative forms using Snowball.

(MRLs) are a challenge for standard embedding models because many inflectional forms are rare or absent even in a large corpus. For example, a Spanish verb has more than 50 forms, many of which are rarely used. This leads to missing or low quality embeddings for such inflectional forms, even for otherwise frequent verbs, i.e., sparsity is a problem. For Finnish and Turkish, this is even more of a problem. Therefore, we propose to compute normalized embeddings instead of embeddings for surface/inflectional forms (referred to as *forms* throughout the rest of the chapter): STem EMbeddings (STEM) for word stems and LemmA eMBeddings (LAMB) for lemmata.

Stemming is a heuristic approach to reducing form-related sparsity issues. Based on simple rules, forms are converted into their stem. However, often the forms of one word are converted into several different stems. For example, present indicative forms of the German verb "brechen" (to break) are mapped to four different stems (see Table 6.1.1). A more principled solution is lemmatization. Lemmatization unites many individual forms, many of which are rare, in one equivalence class, represented by a single lemma. Stems and equivalence classes are more frequent than each individual form. As we will show, this successfully addresses the sparsity issue.

Both methods can learn high-quality semantic representations for rare forms and thus are most beneficial for MRLs as we show below. Moreover, less training data is required to train lemma embeddings of the same quality as form embeddings. Alternatively, we can train lemma embeddings that have the same quality but fewer dimensions than form embeddings, resulting in more efficient applications.

If an application such as parsing requires inflectional information, then stem and lemma embeddings may not be a good choice since they do not contain such information. However, many NLP applications are semantic and for them inflectional information may not even be necessary. For example, most word similarity benchmarks (e.g., MEN (Bruni et al., 2014)) only evaluate semantic similarity, which is largely independent of inflectional morphology. The same is true for polarity classification as we show in this study.

Our contributions in this Chapter are the following:

1. We introduce the normalized embeddings Stem and Lamb and show their usefulness on different tasks for five languages. Although lemmatization is not new and used in other domains, e.g., in information retrieval, only few studies in the word representation domain make use of it (e.g., Melamud et al., (2014), Köper et al., (2015)). This is probably due to the fact that the majority of research is done on English, where sparsity is less of a problem. This chapter is the first study that comprehensively compares stem/lemma-based with form-based embeddings for MRLs.

2. We show the advantage of normalization on word similarity benchmarks. Normalized embeddings yield better performance for MRL languages on most datasets (6/7 for German and 2/2 for Spanish).

3. We propose a new intrinsic relatedness evaluation based on WordNet graphs and publish datasets for five languages. On this new evaluation, Lamb outperforms form-based baselines by a big margin.

4. Stem and Lamb outperform baselines on polarity classification for Czech and English.

5. We show that Lamb embeddings are efficient in that they are high-quality for small training corpora and small dimensionalities.

This chapter is structured as follows. Section 6.2 describes the two normalization methods we use in this study, stemming and lemmatization, and how they are applied. In Section 6.3 we conduct three experiments, word similarity, word relations, and polarity classification. An analysis of the benefits of morphological normalization is given in Section 6.4. Section 6.5 gives an overview over related work and Section 6.6 concludes this chapter.

6.2 stem/lemma creation

The main hypothesis of this work is that normalization addresses sparsity issues, especially for MRLs, because although a particular word form might not have been seen in the text, its stem or lemma is more likely to be known.

Stemmers are available for many languages. Especially Snowball,[1] a "string processing language designed for creating stemming algorithms" is widely used and covers all languages from our experiments. Since it is a rule-based approach, training data is not necessary.

We use the term *lemma* as that surface form that represents the set of word forms that belong to one equivalence class. Lemmata allow the mapping of words to lexical resources. For lemmatization we use the pipeline version of the freely available, high-quality lemmatizer Lemming (Müller et al., 2015). Since it is a language-independent token-based lemmatizer it is especially suited for our multi-lingual experiments. Moreover, it reaches state-of-the-art performance for the five languages that we study.

Lemming's pipeline annotates tokens in context by first running the morphological tagger MarMoT (Müller et al., 2013). It then creates a set of lemma candidates for each token by applying a set of edit trees (Chrupała, 2008). These edit trees encode a sequence of replacement operations such as remove "s" (as in "walks" - "walk") or replace "oo" with "ee" (as in "feet" - "foot") and are used to convert the current token into a lemma candidate. The resulting candidate set is then scored in a log-linear model using a number of features such as the edit tree index, the aligned replacement operations, and features that test whether the resulting lemma occurs in a dictionary or has a high unigram count in some raw text corpus. We train the pipeline using the Penn Treebank (Marcus et al., 1993) for English, SPMRL 2013 shared task data (Seddah et al., 2013) for German and Hungarian, and CoNLL 2009 (Hajič et al., 2009) datasets for Spanish and Czech. We additionally use a unigram list extracted from Wikipedia datasets and the Aspell dictionary of each language.[2]

6.3 EXPERIMENTS

We conduct two intrinsic and one extrinsic evaluation. The two intrinsic evaluations compute word similarities based on either forms, stems, or lemmata and demonstrate that stem- and lemma-based similarities are superior to form-based similarities. The extrinsic evaluation is polarity classification for Czech and English. We show that a system based on STEM and LAMB are consistently better than a system based on form embeddings.

6.3.1 *Word Similarity*

One popular way of evaluating embedding quality is through word similarity. A model needs to assign a similarity score to predefined

[1] snowball.tartarus.org

[2] ftp://ftp.gnu.org/gnu/aspell/dict

lang.	dataset	pairs	reference
de	Gur30	29	Gurevych, (2005)
	Gur350	350	Gurevych, (2005)
	Gur65	65	Gurevych, (2005)
	MSL	999	Leviant and Reichart, (2015)
	MWS	350	Leviant and Reichart, (2015)
	WS	280	Köper et al., (2015)
	ZG222	222	Zesch and Gurevych, (2006)
en	MC	30	Miller and Charles, (1991)
	MEN	1000	Bruni et al., (2014)
	RG	65	Rubenstein and Goodenough, (1965)
	RW	2034	Luong et al., (2013)
	SL	999	Hill et al., (2014)
	WS	353	Finkelstein et al., (2002)
es	MC	30	Hassan and Mihalcea, (2009)
	WS	352	Hassan and Mihalcea, (2009)

table 6.3.1: word similarity datasets Word similarity datasets for all three evaluated languages, with abbreviation, number of word pairs, and reference.

word pairs. Given a pair of words (m, n) and a set of embeddings E we compute their similarity as cosine similarity:

$$\begin{aligned} \mathrm{sim}_E(m, n) &= \cos(E_m, E_n) \\ &= \frac{E_m \cdot E_n}{\parallel E_m \parallel \cdot \parallel E_n \parallel} \end{aligned} \tag{6.3.1}$$

where E_m and E_n are the embeddings of m and n respectively. For word pairs where at least one word is unknown to the model we assign $\mathrm{sim}_E(m, n) := 0$. After computing the similarities for all word pairs, Spearman correlation is computed between these similarity scores and human-based judgments. This is done for three sets of embeddings, form embeddings E^F, Stem E^S, and Lamb E^L.

For form embeddings E^F, we directly use the embeddings of the word pairs' forms (E^F_m and E^F_n) and compute their similarity. For Stem we use $E^S_{stem(w)}$, where $stem(w)$ is the stem of w. For Lamb we use $E^L_{lemma(w)}$, where $lemma(w)$ is the lemma of w; we randomly select one of w's lemmata if there are several.

We conduct experiments on English (en), German (de), and Spanish (es). All used datasets with their sizes are listed in Table 6.3.1.

LANG.	CORPUS	# TOKENS	# FORMS	# STEMS	# LEMMAS
CZ	Wikipedia	83M	1461K	873K	869K
DE	COW	7973M	1335K	1059K	1104K
	Wikipedia	609M	8223K	6669K	6876K
EN	Wikipedia	1779M	7741K	7092K	7403K
ES	COW	3681M	373K	274K	229K
	Wikipedia	396M	6395K	5823K	6082K
HU	Wikipedia	85M	2710K	1563K	1557K

TABLE 6.3.2: SIZES OF TRAINING CORPORA Sizes of training corpora with number of tokens, and number of form, stem, and lemma types. The numbers for both COW corpora consider only types that occur at least 50 times.

For good performance, high-quality embeddings, trained on large corpora, are required. Hence, the training corpora for German and Spanish are taken from COW14 (Schäfer, 2015). Preprocessing includes removal of XML, conversion of HTML characters, lowercasing, stemming using Snowball, and lemmatization using Lemming. We use the entire Spanish corpus (3.7 billion tokens), but cut the German corpus to approximately 8 billion tokens to be comparable to Köper et al., (2015) (Table 6.3.2). We train CBOW models (Mikolov, Chen, et al., 2013) for forms, stems, and lemmata using word2vec[3] with the following settings: 400 dimensions, symmetric context of size 2 (no dynamic window), 1 training iteration, negative sampling with 15 samples, a learning rate of 0.025, minimum count of words of 50, and a sampling parameter of 10^{-5}. CBOW is chosen, because it trains much faster than skip-gram.

Since the morphology of English is rather simple we do not expect STEM and LAMB to reach or even surpass highly optimized systems on any word similarity dataset (e.g., Bruni et al., (2014)). Therefore, for practical reasons we use a smaller training corpus, namely the preprocessed and tokenized Wikipedia dataset of Müller and Schütze, (2015) (cf. Table 6.3.2).[4] Embeddings are trained with the same settings (using 5 iterations instead of only 1, due to the smaller size of the corpus: 1.8 billion tokens).

Results

Table 6.3.3 shows the results. Although English has a simple morphology, LAMB improves over form performance on MEN and SL. A tie is

[3] code.google.com/p/word2vec/

[4] cistern.cis.lmu.de/marmot/naacl2015

achieved on RW. These are the three largest English datasets, giving a more reliable result. Both models perform comparably on WS. Here, STEM is ahead by 1 point. Forms are better on the small datasets MC and RG, where a single word pair can have a large influence on the result. Because of the simple morphology of English, STEM/LAMB do not outperform forms or only by a small margin and thus they cannot compete with highly optimized state-of-the-art systems such as Baroni et al., (2014). Their performances are higher on some of the datasets for the *best* of 48 different parameter configurations. Our results are in the range of their results showing comparability of our *untuned* results.

On German, both STEM and LAMB perform better on all datasets except WS. We set the new state-of-the-art of 0.79 on Gur350 (compared to 0.77, Szarvas et al., (2011)) and 0.39 on ZG (compared to 0.25, Botha and Blunsom, (2014)); 0.83 on Gur65 (compared to 0.79, Köper et al., (2015)) is the best performance of a system that does not need additional knowledge bases (cf. Navigli and Ponzetto, (2012) and Szarvas et al., (2011)).

LAMB's results on Spanish are equally good. 0.82 on MC and 0.58 on WS are again the best performances of a system not requiring an additional knowledge base (cf. Navigli and Ponzetto, (2012)). The best performance before was 0.64 for MC and 0.50 for WS (both Hassan and Mihalcea, (2009)). STEM cannot improve over form embeddings, showing the difficulty of Spanish morphology.

To establish comparability of the models, we also report the Spearman correlation only for those word pairs that are covered by all models' vocabularies. Table 6.3.4 lists the results. The results change only slightly. STEM looses slightly on WS (de), but gains on ZG (de). LAMB looses slightly on MWS (de), but also gains on ZG (de) and on RW (en).

6.3.2 *Word Relations*

The second intrinsic evaluation addresses the problem that word similarity benchmarks are not available for many languages and are expensive to create. To remedy this situation, we create word similarity benchmarks that leverage WordNets, which are available for a great number of languages.

Generally, a representation is deemed good if words related by a lexical relation in WordNet – synonymy, hyponymy etc. – have high cosine similarity with this representation. Since the gold standard necessary for measuring this property of a representation can be automatically derived from a WordNet, we can create very large similarity benchmarks with up to 50K lemmata for the five languages we investigate: Czech, English, German, Hungarian, and Spanish.

We view each WordNet as a graph whose edges are the lexical relations encoded by the WordNet, e.g., synonymy, antonymy, and hyponymy. We then define $\mathcal{L}$ as the set of lemmata in a WordNet and

LANG.	DATASET	FORM	STEM	LAMB	COVERAGE
DE	Gur30	0.76	**0.83**	0.80	29, 29, 29
	Gur350	0.74	**0.79**	**0.79**	336, 340, 339
	Gur65	0.80	**0.83**	0.82	65, 65, 65
	MSL	0.44	0.44	**0.47**	994, 995, 995
	MWS	0.60	0.61	**0.62**	348, 350, 350
	WS	**0.72**	**0.72**	0.71	279, 280, 280
	ZG	0.36	0.38	**0.39**	200, 207, 208
EN	MC	**0.82**	0.77	0.80	30, 30, 30
	MEN	0.72	0.73	**0.74**	1000, 1000, 1000
	RG	**0.82**	0.79	0.79	65, 65, 65
	RW	**0.47**	**0.47**	**0.47**	1613, 1947, 1819
	SL	0.42	0.38	**0.43**	998, 999, 999
	WS	0.63	**0.64**	0.63	353, 353, 353
ES	MC	0.70	0.69	**0.82**	30, 30, 30
	WS	0.54	0.54	**0.58**	350, 352, 352

TABLE 6.3.3: WORD SIMILARITY RESULTS FOR FULL VOCABULARY Spearman correlation (ρ) for single models on the full vocabularies of all models. Coverage shows the number of word pairs that are known by the respective model. Bold numbers are the best performance per row.

LANG.	DATASET	FORM	STEM	LAMB	COVERAGE
DE	Gur30	0.76	**0.83**	0.80	29
	Gur350	0.74	**0.79**	**0.79**	336
	Gur65	0.80	**0.83**	0.82	65
	MSL	0.44	0.44	**0.47**	994
	MWS	0.60	**0.61**	**0.61**	348
	WS	**0.72**	0.71	0.71	279
	ZG	0.36	0.40	**0.41**	200
EN	MC	**0.82**	0.77	0.80	30
	MEN	0.72	0.73	**0.74**	1000
	RG	**0.82**	0.79	0.79	65
	RW	0.47	0.47	**0.48**	1613
	SL	0.42	0.38	**0.43**	998
	WS	0.63	**0.64**	0.63	353
ES	MC	0.70	0.69	**0.82**	30
	WS	0.54	0.54	**0.58**	350

TABLE 6.3.4: WORD SIMILARITY RESULTS FOR VOCABULARY INTERSECTION Spearman correlation (ρ) for single models on the intersected vocabularies of all models. Coverage shows the number of word pairs that are known by all models. Bold numbers are the best performance per row.

the distance $d(l, l')$ between two lemmata l and l' as the length of the *shortest path* connecting them in the graph. The k-neighborhood $\mathcal{N}^k(l)$ of l is the set of lemmata l' that have distance k or less, excluding l: $\mathcal{N}^k(l) := \{l' | d(l, l') \leq k, l \neq l'\}$. The rank of l for an embedding set E is defined as:

$$\text{rank}_E^k(l) := \underset{i}{\text{argmin}}\ l_i \in \mathcal{N}^k(l) \tag{6.3.2}$$

where l_i is the lemma at position i in the list of all lemmata in the WordNet, ordered according to cosine similarity to l in descending order. In other words, $\text{rank}_E^k(l)$ computes the rank of the word from the neighborhood with the most similar embedding to l.

We restrict $i \in [1, 10]$ and set $k = 2$ for all experiments in this chapter. We omit the indexes k and E when they are clear from context.

To measure the quality of a set of embeddings we compute the Mean Reciprocal Rank (MRR) on the rank results of all lemmata:

$$\text{MRR}_E = \frac{1}{|\mathcal{L}|} \sum_{l \in \mathcal{L}} \frac{1}{\text{rank}_E(l)} \tag{6.3.3}$$

We compute the MRR only based on those examples the model returns valid neighbors for, i.e., $\text{rank}_E^k \leq 10$ for at least one element in the k-neighborhood (since $i \in [1, 10]$). We denote examples for which a model does not return any valid neighbor as *invalid*.

We create large similarity datasets for five languages: Czech (cz), English (en), German (de), Hungarian (hu), and Spanish (es) by extracting all lemmata from the WordNet version of the respective language. For English and Spanish we use the preprocessed WordNets from the Open Multilingual WordNet Bond and Paik, (2012). We use the Czech and Hungarian WordNets Miháltz et al., (2008) and PALA and SMRZ, (2004) and GermaNet Hamp and Feldweg, (1997) for German. We keep all lemmata that have a known form in the form embeddings and that exist in the lemma embeddings. Moreover, we filter out all synsets that contain only one lemma and discard all multiword phrases (e.g., there are 68082 in the English WordNet, such as "real time"). The split into development and test sets is done in a way that the distribution of synset sizes (i.e., the number of lemmata per synset) is nearly equal in both sets. The number of lemmata in our evaluation sets can be found in Table 6.3.3. For more insight, we report results on all Part-of-Speech (POS), as well as separately for nouns (n), verbs (v), and adjectives (a). Note that the all-POS setting can include further POS, depending on the WordNet. Moreover, some lemmata occur in multiple POS. The datasets are made publicly available.

We propose the following models for the embeddings evaluation. For form embeddings we compare three different strategies, a *realistic* one, an *optimistic* one, and a lemma approximation strategy. In the realistic strategy (*form real*), given a query lemma we randomly sample a form, for which we then compute the k-neighborhood. If the neighbors contain multiple forms of the same equivalence class, we exclude the

lang.	set	all	a	n	v
cz	dev	9694	852	6436	2315
	test	9763	869	6381	2433
de	dev	51682	6347	40674	5018
	test	51827	6491	40623	5085
en	dev	44448	9713	30825	5661
	test	44545	9665	30736	5793
es	dev	12384	1711	8634	1989
	test	12476	1727	8773	1971
hu	dev	19387	1953	15268	2057
	test	19486	1928	15436	2011

table 6.3.5: number of lemmata in wordnet datasets Size of the WordNet datasets in terms of number of lemmata. They are separated by language and POS and split into development and test set.

repetitions and use the next neighbors instead. For instance, if *house* is already a neighbor, then *houses* will be skipped. The optimistic strategy (*form opt*) works similarly, but uses the embedding of the most frequent surface form of a lemma. This is the upper bound a form model can reach, which already requires information about lemma and surface form counts. As a baseline lemma approximation strategy, we sum up all surface form embeddings that belong to one equivalence class (*form sum*). For Stem we repeat the same experiments as described for forms, leading to *stem real*, *stem opt*, and *stem sum*.

For embeddings training, Wikipedia comes as a natural choice as corpus, because it is available for many languages. Therefore, we use the preprocessed and tokenized Wikipedia datasets of Müller and Schütze, (2015) and annotate them with stems using Snowball and with lemmata using Lemming. The resulting corpora sizes are listed in Table 6.3.2.

We train 50-dimensional skip-gram embeddings Mikolov, Chen, et al., (2013) with word2vec on the original, the stemmed, and the lemmatized corpus, respectively. Embeddings are trained for all tokens, because we need a high coverage; the context size is set to 5, all remaining parameters are left at their default value. We train smaller embeddings than before, because we have more models to train and the training corpora are smaller. This furthermore allows us to train the more performance-hungry skip-gram models.

Results

The MRR results in Table 6.3.6 show that for all languages and for all POS, *form real* has the worst performance among the form models. This comes at no surprise since this model does barely know anything about word forms and lemmata. The *form opt* model improves these results based on the additional information it has access to (the mapping from lemma to its most frequent form). *form sum* – approximating the embedding of the lemma by summing the embeddings of its forms – performs similar to *form opt*. For Czech, Hungarian, and Spanish it is slightly better (or equally good), whereas for English and German there is no clear trend. There is a large difference between these two models on German nouns, with *form sum* performing considerably worse. We attribute this to the fact that many German noun forms are rare compounds and therefore lead to badly trained form embeddings, which summed up do not lead to high quality embeddings either.

Among the stemming models *stem real* also is the worst performing model. We can further see that for all languages and almost all POS, *stem sum* performs worse than *stem opt*. That indicates that stemming leads to many low-frequency stems or many words sharing the same stem. This is especially apparent in Spanish verbs. There, the stemming models are clearly inferior to form models.

Overall, LAMB performs best for all languages and POS types. Most improvements of LAMB are significant. The improvement to the best form-model reaches up to 6 points (e.g., Czech nouns). In contrast to *form sum*, LAMB improves over *form opt* on German nouns. This indicates that the sparsity issue is successfully addressed by LAMB.

In general, morphological normalization in terms of stemming or lemmatization improves the result on all languages, leading to an especially substantial improvement on MRLs. For the morphologically very rich languages Czech and Hungarian, the relative improvement of STEM or LAMB to form-based models is especially high; e.g., Hungarian all: 50%. Moreover, we find that MRLs yield lower absolute performance. This confirms the findings of Köper et al., (2015). Surprisingly, LAMB yields better performance on English despite its simple morphology.

The number of invalid lemmata together with the total number of query lemmata is listed in Table 6.3.7.We can see that STEM retrieves more valid neighbors than form-based models. LAMB retrieves more valid neighbors than all other models.

The low absolute results and the still high number of invalid examples – especially for Hungarian – show that we address a challenging task and that our new evaluation methodology is a good evaluation for new types of word representations.

For further insight, we restrict the nearest neighbor search space (i.e., k-neighborhood) to those lemmata that have the same POS as the query lemma. Note that this restriction does not need additional resources,

		form			Stem			
lang.	POS	real	opt	sum	real	opt	sum	Lamb
cz	a	0.03	0.04	0.05	0.02	0.05	0.05	**0.06**
	n	$0.15^{\ddagger}$	$0.21^{\ddagger}$	$0.24^{\ddagger}$	$0.18^{\ddagger}$	$0.27^{\ddagger}$	$0.26^{\ddagger}$	**0.30**
	v	$0.07^{\ddagger}$	$0.13^{\ddagger}$	$0.16^{\dagger}$	$0.08^{\ddagger}$	$0.14^{\ddagger}$	$0.16^{\ddagger}$	**0.18**
	all	$0.12^{\ddagger}$	$0.18^{\ddagger}$	$0.20^{\ddagger}$	$0.14^{\ddagger}$	$0.22^{\ddagger}$	$0.21^{\ddagger}$	**0.25**
de	a	$0.14^{\ddagger}$	$0.22^{\ddagger}$	$0.25^{\dagger}$	$0.17^{\ddagger}$	0.26	$0.21^{\ddagger}$	**0.27**
	n	$0.23^{\ddagger}$	$0.35^{\ddagger}$	$0.30^{\ddagger}$	$0.28^{\ddagger}$	$0.35^{\dagger}$	$0.33^{\ddagger}$	**0.36**
	v	$0.11^{\ddagger}$	$0.19^{\ddagger}$	$0.18^{\ddagger}$	$0.11^{\ddagger}$	0.22	$0.18^{\ddagger}$	**0.23**
	all	$0.21^{\ddagger}$	$0.32^{\ddagger}$	$0.28^{\ddagger}$	$0.24^{\ddagger}$	$0.33^{\dagger}$	$0.30^{\ddagger}$	**0.34**
en	a	$0.22^{\ddagger}$	$0.25^{\ddagger}$	$0.24^{\ddagger}$	$0.16^{\ddagger}$	$0.26^{\ddagger}$	$0.25^{\ddagger}$	**0.28**
	n	$0.24^{\ddagger}$	$0.27^{\ddagger}$	$0.28^{\ddagger}$	$0.22^{\ddagger}$	**0.30**	$0.28^{\ddagger}$	**0.30**
	v	$0.29^{\ddagger}$	$0.35^{\ddagger}$	**0.37**	$0.17^{\ddagger}$	0.35	$0.24^{\ddagger}$	**0.37**
	all	$0.23^{\ddagger}$	$0.26^{\ddagger}$	$0.27^{\ddagger}$	$0.20^{\ddagger}$	$0.28^{\ddagger}$	$0.25^{\ddagger}$	**0.29**
es	a	$0.20^{\ddagger}$	$0.23^{\ddagger}$	$0.23^{\ddagger}$	$0.08^{\ddagger}$	$0.21^{\ddagger}$	$0.18^{\ddagger}$	**0.27**
	n	$0.21^{\ddagger}$	$0.25^{\ddagger}$	$0.25^{\ddagger}$	$0.16^{\ddagger}$	$0.25^{\ddagger}$	$0.23^{\ddagger}$	**0.29**
	v	$0.19^{\ddagger}$	$0.35^{\dagger}$	0.36	$0.11^{\ddagger}$	$0.29^{\ddagger}$	$0.19^{\ddagger}$	**0.38**
	all	$0.20^{\ddagger}$	$0.26^{\ddagger}$	$0.26^{\ddagger}$	$0.14^{\ddagger}$	$0.24^{\ddagger}$	$0.21^{\ddagger}$	**0.30**
hu	a	$0.02^{\ddagger}$	$0.06^{\ddagger}$	$0.06^{\ddagger}$	$0.05^{\ddagger}$	0.08	0.08	**0.09**
	n	$0.01^{\ddagger}$	$0.04^{\ddagger}$	$0.05^{\ddagger}$	$0.03^{\ddagger}$	**0.07**	$0.06^{\ddagger}$	**0.07**
	v	$0.04^{\ddagger}$	$0.11^{\ddagger}$	$0.13^{\ddagger}$	$0.07^{\ddagger}$	$0.14^{\ddagger}$	0.15	**0.17**
	all	$0.02^{\ddagger}$	$0.05^{\ddagger}$	$0.06^{\ddagger}$	$0.04^{\ddagger}$	$0.08^{\ddagger}$	$0.07^{\ddagger}$	**0.09**

table 6.3.6: word relation results on the unfiltered test set MRR results per language and POS types for all models on the test set. Significance (sign test) is compared to Lamb with ‡: $p = 0.01$, †: $p = 0.05$. Bold is the best performance per row.

			form			Stem			
lang.	POS	total	real	opt	sum	real	opt	sum	Lamb
cz	a	869	832	814	790	834	792	789	**778**
	n	6381	4611	3869	3643	4246	3403	3455	**3106**
	v	2433	2053	1794	1721	2020	1763	1706	**1629**
	all	9763	7530	6542	6218	7207	6020	6014	**5576**
de	a	6491	4909	4200	3954	4688	3883	4045	**3820**
	n	40623	25680	19088	20947	22774	18435	18774	**17951**
	v	5085	4043	3414	3370	4026	3193	3302	**3108**
	all	51827	34328	26575	28126	31311	25390	25992	**24776**
en	a	9665	6297	5925	5955	7083	5766	5614	**5562**
	n	30736	18689	16982	16384	19038	15888	16321	**15698**
	v	5793	3085	2542	**2292**	3870	2416	2871	2311
	all	44545	27825	25604	24887	29674	24448	25028	**23903**
es	a	1727	1207	1108	1114	1473	1129	1157	**1037**
	n	8773	5668	5100	5056	6275	5075	5264	**4598**
	v	1971	1202	813	799	1469	934	1102	**775**
	all	12476	8185	7127	7078	9304	7257	7622	**6530**
hu	a	1928	1841	1720	1731	1758	1638	1645	**1615**
	n	15436	15096	14372	14040	14569	13759	13886	**13580**
	v	2011	1865	1621	1577	1755	1522	1479	**1424**
	all	19486	18823	17777	17413	18173	17003	17079	**16679**

table 6.3.7: number of invalid results on the unfiltered test set Number of invalid lemmata, i.e., where a model returns a k-neighborhood with all items having $\text{rank}_E^k > 10$, per language and POS for all models on the test set (smaller is better). Bold is the best number per row.

		form			Stem			
lang.	POS	real	opt	sum	real	opt	sum	Lamb
cz	a	0.03‡	0.05†	0.07	0.04†	0.08	0.08	**0.09**
	n	0.17‡	0.23‡	0.26‡	0.20‡	0.29‡	0.28‡	**0.32**
	v	0.09‡	0.15‡	0.17‡	0.09‡	0.17†	0.18	**0.20**
de	a	0.17‡	0.25‡	0.27‡	0.23‡	**0.33**	**0.33**	**0.33**
	n	0.24‡	0.36‡	0.31‡	0.28‡	0.36	0.35‡	**0.37**
	v	0.13‡	0.20‡	0.21‡	0.13‡	0.24‡	0.23‡	**0.26**
en	a	0.25‡	0.28‡	0.28‡	0.18‡	0.29‡	**0.32**	0.31
	n	0.25‡	0.28‡	0.29‡	0.23‡	0.31†	0.31‡	**0.32**
	v	0.33‡	0.39‡	0.42‡	0.21‡	0.42†	0.39‡	**0.44**
es	a	0.21‡	0.25‡	0.26‡	0.10‡	0.26‡	0.26‡	**0.30**
	n	0.22‡	0.26‡	0.27‡	0.17‡	0.27‡	0.26‡	**0.30**
	v	0.22‡	0.36‡	0.36‡	0.16‡	0.36‡	0.33‡	**0.42**
hu	a	0.04‡	0.08‡	0.08‡	0.06‡	**0.12**	0.11	**0.12**
	n	0.01‡	0.04‡	0.05‡	0.04‡	**0.07**†	0.06‡	**0.07**
	v	0.05‡	0.13‡	0.14‡	0.07‡	0.15‡	0.16†	**0.19**

table 6.3.8: word relation results on the filtered test set
MRR results per language and POS type for all models on the test set. The *k*-neighborhood is restricted to lemmata of the same POS as the query lemma. The Significance (sign test) is compared to Lamb with ‡: $p = 0.01$, †: $p = 0.05$. Bold is the best performance per row.

because the lemmatizer yields POS tags as well. The general findings in Table 6.3.8 are similar to the unrestricted experiment: Normalization leads to superior results. The *form real* and *stem real* models yield the lowest performance. *Form opt* improves the performance and *form sum* is better on average than *form opt*. *Stem sum* can rarely improve on *stem opt*. The best stemming model most often is better than the best form model. Lamb can benefit more from the POS type restriction than the form models. The distance to the best form model generally increases, especially on German adjectives and Spanish verbs. In all cases except on English adjectives, Lamb yields the best performance. Again, in almost all cases Lamb's improvement over the form-models is significant.

Except for Spanish, the best Stem model retrieves more valid examples than the best form model (Table 6.3.9). Overall, Lamb is ahead of

			FORM			STEM			
LANG.	POS	TOTAL	REAL	OPT	SUM	REAL	OPT	SUM	LAMB
CZ	a	869	814	789	761	799	743	747	**735**
	n	6381	4377	3652	3416	3997	3180	3167	**2936**
	v	2433	1947	1704	1659	1921	1643	1600	**1561**
DE	a	6491	4615	3866	3632	4067	**3267**	3277	3272
	n	40623	25169	18590	20261	22503	17854	17979	**17546**
	v	5085	3758	3207	3120	3782	2926	3009	**2821**
EN	a	9665	5890	5482	5455	6735	5338	**5111**	5175
	n	30736	18255	16630	15903	18849	15402	15608	**15213**
	v	5793	2603	2163	1988	3228	**1849**	2071	1867
ES	a	1727	1142	1036	1021	1401	1039	1039	**964**
	n	8773	5450	4891	4840	6139	4882	4941	**4479**
	v	1971	1140	771	790	1213	746	789	**673**
HU	a	1928	1784	1635	1625	1708	1521	1539	**1513**
	n	15436	15091	14357	14034	14472	13675	13807	**13500**
	v	2011	1814	1533	1505	1734	1483	1432	**1356**

TABLE 6.3.9: NUMBER OF INVALID RESULTS ON THE FILTERED TEST SET Number of invalid lemmata, i.e., where a model returns a k-neighborhood with all items having $\text{rank}_E^k > 10$, per language and POS for all models on the test set (smaller is better) The k-neighborhood is restricted to lemmata of the same POS as the query lemma. Bold is the best number per row.

the other models in almost all cases. It is no surprise that the models retrieve more valid examples compared to the unrestricted experiment (cf. Table 6.3.7).

6.3.3 *Polarity Classification*

Our first two evaluations were intrinsic and show that the normalized embeddings are of high quality. We now analyze the influence of morphological normalization on polarity classification. For that we conduct similar experiments as in Chapter 5, namely polarity classification on English Twitter tweets. LINGCNN (cf. Section 5.2) is used for classification. We use the features explained in Section 5.2, with the following extensions:

DATASET	TOTAL	POSITIVE	NEGATIVE	NEUTRAL
Twitter 2015 train	9845	3636	1535	4674
Twitter 2015 dev	3813	1572	601	1640
Twitter 2015 test	2390	1038	365	987
Czech Film Database (CSFD)	91379	30896	29716	30767

TABLE 6.3.10: POLARITY CLASSIFICATION DATASETS Datasets for the polarity classification experiments with their number of tweets/reviews in total and per polarity class. The Twitter dataset is the same as the one used in Chapter 5.

1. Three new word-level binary sentiment indicators are created for the emoticons from the SentiStrength lexicon. One for each emoticon category of positive, negative, and neutral.

2. Three new sentence-level count features are created accordingly (instead of just one for all polarities as done in Section 5.2.2).

As dataset the SemEval 2015 Twitter data from Table 5.3.1 (for convenience also shown in Table 6.3.10) is reused (Rosenthal et al., 2015). We train LINGCNN using Stochastic Gradient Descent (SGD) with AdaGrad Duchi et al., (2011) and early stopping (maximum number of epochs: 30), batch size = 100, 100 filters each per width of $2 \leq m \leq 5$; k-max pooling with $k = 1$; learning rate $\eta = 0.01$; and ℓ_2 regularization ($\lambda = 5e^{-5}$). We reuse the 50-dimensional Wikipedia embeddings from Section 6.3.2. As training vocabulary we use the 100K most frequent word types of the Wikipedia embeddings.

Since the morphology of English is very simple, another experiment on Czech is performed. The task is classification of Czech movie reviews from the CSFD project (Habernal et al., 2013)) into positive, negative, or neutral (Table 6.3.10). Since there are fewer resources available LINGCNN has fewer features. Linguistic word level features are:

BINARY SENTIMENT INDICATORS We create two features from the SubLex 1.0 sentiment lexicon (Veselovská and Bojar, 2013). One for each of positive and negative. In addition we create three features for the emoticons from the SentiStrength lexicon.

BINARY NEGATION Differently than English, negation in Czech is indicated by the word prefix "ne". Thus, instead of marking words between a negation word and the following punctuation character as negated, we use this prefix instead. We however disregard words with the prefix "nej" as negation indicators, because they indicate superlatives. Exceptions from this rule (i.e., words with "nej" as prefix that are indeed negation indicators) are common negated words such as "nejsi" (Engl. "you are not"). Table 6.3.11 lists the used exceptions.

form	English translation
nejsem	I am not
nejsi	you are not
nejsme	we are not
nejste	you are not (plural)
nejsou	they are not

table 6.3.11: list of czech superlative exceptions List of words having the prefix "nej" but being negations instead of superlatives.

The used linguistic sentence-level features are the same as for English with the exception that the sentiment score features (Section 5.2.2) for the SubLex lexicon are not computed separately based on their POS.

The CSFD dataset is larger than the SemEval dataset. Hence, we choose different hyperparamters for the model training: We use the entire vocabulary of the Wikipedia embeddings. We use 200 filters each for filter widths of $3 \leq m \leq 6$ and set k-max pooling to $k = 5$. The other hyperparameters remain the same as for the English experiment.

For both languages we compare three experimental conditions: using forms, Stem, and Lamb. In order to establish comparability and analyzing only the effect of embeddings, for all three model variants we compute the linguistic features based on the original dataset. Only the embeddings part of the model (matrix P in Section 5.2.1) is changed based on the stemmed and lemmatized dataset.

Results

On the SemEval data, Stem performs comparably to the form model. Lamb improves the results over form and stem in terms of both accuracy and macro F_1 (cf. Table 6.3.12).[5] Hence, Lamb can still pick up additional information despite the simple morphology of English. This is probably due to better embeddings for rare words. The SemEval 2015 winner Hagen et al., (2015) is a highly domain-dependent and specialized system that we do not outperform.

The lower half of Table 6.3.12 lists the 10-fold cross-validation results (accuracy and macro F_1) on the CSFD dataset. Lamb/Stem results are consistently better than form results.

In the introduction, we discussed that normalization removes inflectional information that is necessary for NLP tasks like parsing. For polarity classification, comparatives and superlatives can be important. Further analysis is necessary to determine whether their normalization

[5] To be comparable with published results we report the macro F_1 of positive and negative classes. Cf. Equation 5.3.1.

LANG.	FEATURES	ACC.	F_1
EN	Hagen et al., (2015)	-	**64.84**
	form	66.78	62.21
	STEM	66.95	62.06
	LAMB	**67.49**	63.01
CZ	Brychcin and Habernal, (2013)	-	**81.53**
	form	80.86	80.75
	STEM	**81.51**	81.39
	LAMB	81.21	81.09

TABLE 6.3.12: POLARITY CLASSIFICATION RESULTS Accuracy and macro F_1 performance of state-of-the-art model and three versions of LINGCNN. For English, macro F_1 is computed for positive and negative classes. For Czech, it is computed on all three classes. Bold is best per language and column.

hurts in our experiments. However, note that we evaluate on polarity only, not on valence, i.e., the magnitude of positivity and negativity.

Furthermore, the following example shows how sparsity is successfully addressed by LAMB: "popis a název zajmavý a film je taková filmařská prasárna ." ("Description and title are interesting, but it is bad film-making."). The underlined words "zajmavý" (interesting) and "prasárna" (bad, smut) are unknown to the form model, because they do not occur in the embeddings training file. The latter however is known to LAMB, which is then able to classify this example correctly as negative.

6.4 ANALYSIS

Normalized embeddings deal better with sparsity than form embeddings. In this section, we demonstrate two additional benefits of LAMB based on its robustness against sparsity. First, we train lower-dimensional lemma embeddings and still reach the same performance as form embeddings with higher dimensionality. Second we need less training data for embeddings to reach the same performance.

6.4.1 *Embedding Size*

We now show that LAMB can train embeddings with fewer dimensions on the same amount of data and still reach the same performance as larger form embeddings. We repeat the word relation experiments of Section 6.3.2 (all POS) and train all models with embeddings sizes 10, 20, 30, 40, and 50 for Spanish. We choose Spanish because it has richer

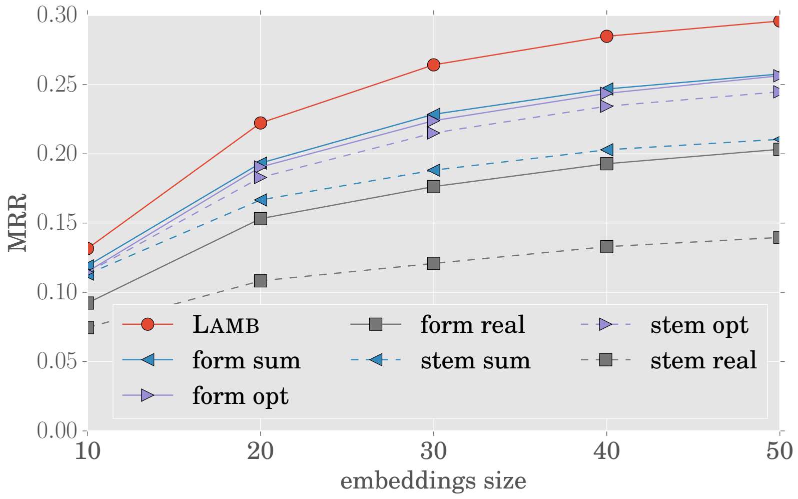


FIGURE 6.4.1: EMBEDDING SIZE ANALYSIS MRR of the word relation experiment on Spanish (all POS) with respect to embeddings size.

morphology than English and more training data than Czech and Hungarian.

Figure 6.4.1 depicts the MRR results of all models with respect to embeddings size. The relative ranking of form models is *real* < *opt* < *sum*. That comes from the additional information the more complex models have access to. All stemming models reach lower performance than their form counterparts (similar to results in Table 6.3.6). That suggests that stemming is not a proper alternative to correctly dealing with Spanish morphology. The relative ranking of stem models is *real* < *sum* < *opt*. LAMB reaches higher performance than *form real* with already 20 dimensions. The 30 dimensional LAMB model is better than all other models. Thus, we can create lower-dimensional lemma embeddings that are as good as higher-dimensional form embeddings; this has the benefits of reducing the number of parameters in models using these embeddings and of reducing training times and memory consumption.

6.4.2 *Corpus Size*

Our second hypothesis is that less training data is necessary to train good embeddings. We create 10 training corpora consisting of the first k percent, $k \in \{10, 20, \ldots, 100\}$, of the randomized Spanish Wikipedia corpus. With these 10 subcorpora we repeat the word relation experiments of Section 6.3.2 (all POS). As query lemmata, we use the lemmata from before that exist in all subcorpora.

Figure 6.4.2 shows that the relative ranking among the models is the same as before. This time however, *form sum* yields better performance than *form opt*, especially when little training data is available. Recall

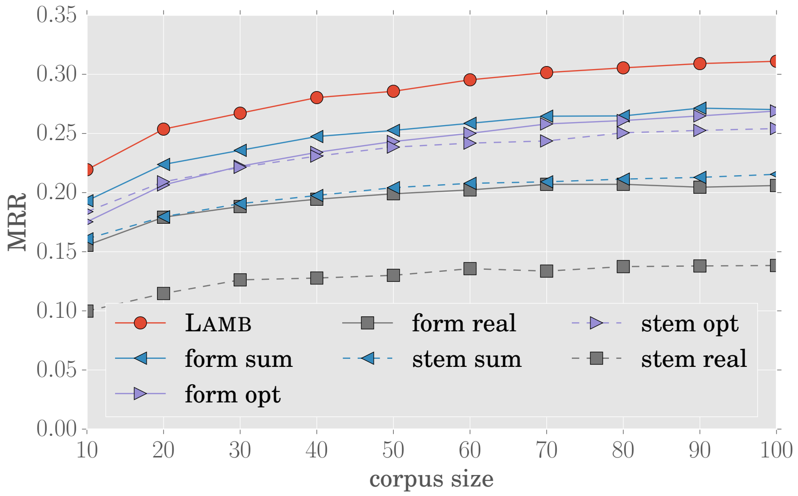


FIGURE 6.4.2: CORPUS SIZE ANALYSIS MRR of the word relation experiment on Spanish (all POS) with respect to corpus size.

that *form opt* is similar to an approach that is used in most systems that have embeddings, which just use the available surface forms.

The stemming models again are inferior to their form counterparts. Only *stem opt* is able to reach similar performance than *form opt*. Lamb always reaches higher performance than *form real*, even when only 10% of the training corpus is used. With 30% of the training corpus, Lamb surpasses the performance of the other models. Again, by requiring less than 30% of the training data, embedding training becomes much more efficient. Furthermore, in low-resource languages that lack the availability of a large homogeneous corpus, Lamb can still be trained successfully.

6.5 RELATED WORK

There has been a large number of studies on English, a morphologically simple language, that show that the effect of normalization, in particular stemming, is different for different applications. For instance, Karlgren and Sahlgren, (2001) analyze the impact of morphological analysis on creating word representations for synonymy detection. They compare several stemming methods. Bullinaria and Levy, (2012) use stemming and lemmatization before training word representations. The improvement of morphological normalization in both studies is moderate in the best case. Melamud et al., (2014) compute lemma embeddings to predict related words given a query word. They do not compare form and lemma representations.

A finding about English morphology does not provide insight into what happens with the morphology of an MRL. In this chapter we use English to provide a data point for morphologically poor languages. Although we show that normalization for embeddings increases performance significantly on some applications – a novel finding to the best of our knowledge – morphologically simple languages (for which normalization is expected to be less important) are not the main focus of the chapter. Instead, MRLs are the main focus. For these, we show large improvements on several tasks.

Recently, Köper et al., (2015) compared form and lemma embeddings on English and German focusing on morpho-syntactic and semantic relation tasks. Generally, they found that lemmatization has limited impact. We extensively study MRLs and find a strong improvement on MRLs when using normalization, on intrinsic as well as extrinsic evaluations.

Synonymy detection is a well studied problem in the NLP community (Baroni and Bisi, 2004; Grigonyte et al., 2010; Ruiz-Casado et al., 2005; Turney et al., 2003; Turney, 2001). Rei and Briscoe, (2014) classify hyponomy relationships through embedding similarity. Our premise is that semantic similarity comprises all of these relations and more. Our ranking-based word relation evaluation addresses this issue. Similar to Melamud et al., (2014), our motivation is that, in contrast to standard word similarity benchmarks, large resources can be automatically generated for any language with a WordNet. This is also exploited by Tsvetkov et al., (2015). Their intrinsic evaluation method requires an annotated corpus, e.g., annotated with WordNet supersenses. Our approach requires only the WordNet.

An alternative strategy of dealing with data sparsity is presented by Soricut and Och, (2015). They compute morphological features in an unsupervised fashion in order to construct a form embedding by the combination of the word's morphemes. We address scenarios (such as polarity classification) in which morphological information is less important, thus form embeddings are not needed.

6.6 CONCLUSION

We have presented Stem and Lamb, embeddings based on stems and lemmata. In three experiments we have shown the superiority compared to commonly used form embeddings. Especially (but not only) on MRLs, where data sparsity is a problem, both normalized embeddings perform better than form embeddings by a large margin. In a new challenging WordNet-based experiment we have shown four methods of adding morphological information (*opt*, *sum*, Stem, Lamb). Here, Lamb is the best of the proposed ways of using morphological information, consistently reaching (much) higher performance. Stem methods are not consistently better, indicating that the more principled way of normalization as done by Lamb is to be preferred. The datasets are published at http://www.cis.lmu.de/ebert.

Our analysis shows that by using Lamb, fewer embedding dimensions or less embedding training data is required to reach the same performance as with form embeddings, making it appealing for under-resourced languages.

The use of linguistic morphological analyzers may not be justified for morphologically poor languages like English. The cost/benefit trade-off in that case is in favor of purely data-driven methods. This however is different for MRLs. Especially, since a finding about English morphology does not indicate what happens with the morphology of an MRL. Since morphological analyzers are becoming available for more and more languages, we show that better results can be obtained almost for free.

6.7 FUTURE WORK

The following points are possible future directions of this work:

- The lemma approximation strategy *form sum* in Section 6.3.1 and Section 6.3.2 can be improved by using a weighted average, leading to another interesting baseline.

- One possible extension of this work would be to use normalization only on infrequent word forms. This would lead to high quality embeddings for a word's most frequent form(s) and a single embedding covering all infrequent word forms.

- As pointed out in Section 6.3.3 future work needs to analyze the exact effect of morphology on Sentiment Analysis (SA) applications. Whereas for polarity classification superlatives, etc. do not seem to require special handling, for valence or fine-grained polarity classification (more than two categories of polarity) the opposite might apply.

- We hypothesize that morphological information is not required for some Natural Language Processing (NLP) applications. For instance, the TOEFL dataset (Landauer and Dumais, 1997) requires to find the most similar word out of four choices given a query word. Similarly, in the MSR Sentence Completion Challenge the task is to choose one out of five given words to be filled in into a sentence depending on context words (Zweig and Burges, 2011). Both these tasks are semantic in nature and may not require morphology.

- We compute all linguistic features in the polarity classification experiment based on the original unstemmed and unlemmatized dataset. For comparability this is beneficial, allowing to see the effect of the different embeddings only. In terms of performance this might be counter-productive, because the lemmatized dataset might match more entries of the sentiment lexicons. Conducting such an experiment might improve the results in Table 6.3.12.

- In order to have a better understanding of the data and the differences among the 5 used languages, the analysis in Section 6.4 can be conducted on all languages and on all three presented tasks.

7 ULTRADENSE SENTIMENT REPRESENTATIONS

> This chapter covers work already published at international peer-reviewed conferences. The relevant publication is Rothe et al., (2016).
> I was the primary contributor to all experimental work on polarity classification described in Section 7.4.4. Sascha Rothe developed the formalization of ultradense embeddings and was the primary contributor to the experimental work on concreteness, frequency, and association strength. The last author of the publication acted as advisor.

As we saw earlier, embeddings are *generic* representations that are useful for many NLP tasks. In this chapter, we want to use a new method of computing sentiment-specific word embeddings for polarity classification. The method, Densifier, learns an orthogonal transformation of the embedding space that focuses the information relevant for a task in an *ultradense subspace* of a dimensionality that is smaller by a factor of 100 than the original space. We show that ultradense embeddings generated by Densifier reach state of the art on a lexicon creation task in which words are annotated with three types of lexical information – sentiment, concreteness, and frequency. On the SemEval 2015 Task 10B polarity classification task we show that no information is lost when the ultradense subspace is used, but training is an order of magnitude more efficient due to the compactness of the ultradense space.

7.1 INTRODUCTION

Embeddings are a useful building block for many tasks, including word similarity (cf., Chapter 6 and e.g., Pennington et al., (2014)), Named Entity Recognition (NER) (e.g., Collobert et al., (2011)) and Sentiment Analysis (SA) (cf. Chapter 5, Chapter 6, and e.g., Kalchbrenner et al., (2014), Kim, (2014), and Severyn and Moschitti, (2015)). Embeddings are generic, task-independent representations, containing different types of information about a word. It is usually the responsibility of a statistical model to make best use of these generic representations for a specific application like NER or SA. Our hypothesis in this chapter is that the information useful for any given task is contained in an *ultradense subspace*. This chapter describes the method Densifier that is used to identify the ultradense subspace E^u. Given a set of word embeddings, Densifier learns an *orthogonal transformation* of the original space E^o on a task-specific training set. The orthogonality of the transformation can be considered a hard regularizer.

The benefit of this method is that embeddings are most useful if learned on unlabeled corpora and performance-enhanced on a broad array of tasks. This means we should try to keep all information offered by them. Orthogonal transformations "reorder" the space without adding or removing information and preserve the bilinear form, i.e., Euclidean distance and cosine distance. The transformed embeddings concentrate all information relevant for the task in E^u.

The benefits of the ultradense subspace E^u compared to the original space E^o are (i) high-quality and (ii) efficient representations:

1. Densifier moves non-task-related information outside of E^u, i.e., into the orthogonal complement of E^u. As a result, E^u provides higher-quality representations for the task than E^o. For example, noise that could result in overfitting is reduced in E^u compared to E^o.

2. E^u has a dimensionality smaller by a factor of 100 in our experiments. As a result, training statistical models on these embeddings is much faster. These models also have many fewer parameters, thus again helping to prevent overfitting, especially for complex, deep neural networks.

In the most extreme form, ultradense representations – i.e., E^u – have a single dimension. We exploit this for creating lexicons in which words are annotated with lexical information, e.g., with sentiment. Specifically, we create high-coverage lexicons with up to 3 million words (i) for three lexical properties: sentiment, concreteness, and frequency; (ii) for five languages: Czech, English, French, German, and Spanish; (iii) for two domains: Twitter and news, in a domain adaptation setup.

The main advantages of this method of lexicon creation are:

1. We need a training lexicon of only a few hundred words, thus making the method effective for new domains and languages and requiring only a minimal manual annotation effort.

2. The method is applicable to any set of embeddings, including phrase and sentence embeddings. Assuming the availability of a small hand-labeled lexicon, Densifier automatically creates a domain dependent lexicon based on a set of embeddings learned on a large corpus of the domain.

3. While the input lexicon is discrete – e.g., positive (+1) and negative (-1) polarity – the output lexicon is continuous and this more fine-grained assessment is potentially more informative than a simple binary distinction.

We show that lexicons created by Densifier beat the state of the art on SemEval 2015 Task 10E (determining association strength).

Our contribution in this chapter is to use ultradense embeddings for polarity classification on the English SemEval 2015 Task 10B and the Czech Film Database (CSFD) datasets. We show that by using sentiment-focused embeddings that are smaller by a factor of 100 we almost get the same results as with the original vectors. However, the efficiency of the training is much higher.

One of our goals is to make embeddings more interpretable. The work on sentiment, concreteness, and frequency we describe in this chapter is a first step towards a general decomposition of embedding spaces into meaningful, dense subspaces. This would lead to cleaner and more easily interpretable representations – as well as representations that are more effective and efficient.

This chapter is divided into the following sections. Section 7.2 describes the model that creates the ultradense subspace out of generic word representations. In Section 7.3 we show new lexicons and explain how they were created. The resulting ultradense representations are then evaluated in Section 7.4, which is followed by an analysis of the number of subspace dimensions and the size of training resources in Section 7.5. Section 7.6 describes related work and Section 7.7 concludes this chapter.

7.2 MODEL

Let $Q \in \mathbb{R}^{d \times d}$ be an orthogonal matrix that transforms the original word embedding space $E^o \subset \mathbb{R}^{d \times |V|}$ into a space in which certain types of information are represented by a small number of dimensions. Concretely, we learn Q such that the dimensions $D^s \subset \{1, \dots, d\}$ of the resulting space correspond to a word's sentiment information and the $\{1, \dots, d\} \setminus D^s$ remaining dimensions correspond to non-sentiment information. Analogously, the sets of dimensions D^c and D^f correspond to a word's concreteness information and frequency information, respectively. In this chapter, we assume that these properties do not correlate and therefore the ultradense subspaces do not overlap, i.e., $D^s \cap D^c = D^s \cap D^f = D^f \cap D^c = \emptyset$. However, this might not be true for other settings, e.g., sentiment and semantic information.

If $e_w = E^o_w$ with $e_w \in \mathbb{R}^d$ is the original embedding of word w, the transformed representation is Qe_w. We use $*$ as a placeholder for s, c, and f and call $d^* = |D^*|$ the dimensionality of the ultradense subspace of $*$. For each ultradense subspace, we create $P^* \in \mathbb{R}^{d^* \times d}$, an identity matrix for the dimensions in $D^* \subset \{1, \dots, d\}$ and a zero matrix for the residual dimensions. Thus, the ultradense representation $u^*_w = E^u_w$ with $u^*_w \in \mathbb{R}^{d^*}$ of e_w is defined as:

$$u^*_w := P^* Q e_w \tag{7.2.1}$$

7.2.1 *Separating Words of Different Groups*

We assume to have a lexicon resource l in which each word w is annotated for a certain information as either $l^*(w) = +1$ (positive, concrete, frequent) or $l^*(w) = -1$ (negative, abstract, infrequent). We now want to separate words from each other, that have different information, e.g., we want to separate positive from negative words.

Let $\mathcal{L}^*_{\not\sim}$ be a set of word index pairs (v, w) for which $l^*(v) \neq l^*(w)$ holds. We want to maximize:

$$\sum_{(v,w)\in\mathcal{L}^*_{\not\sim}} \|u^*_v - u^*_w\| \tag{7.2.2}$$

Thus, our objective is given by:

$$\underset{Q}{\operatorname{argmax}} \sum_{(v,w)\in\mathcal{L}^*_{\not\sim}} \|P^*Q(e_w - e_v)\| \tag{7.2.3}$$

or, equivalently, by:

$$\underset{Q}{\operatorname{argmin}} \sum_{(v,w)\in\mathcal{L}^*_{\not\sim}} -\|P^*Q(e_w - e_v)\| \tag{7.2.4}$$

subject to Q being an orthogonal matrix.

7.2.2 *Aligning Words of the Same Group*

Another goal is to minimize the distance of two words of the same group. For example, we want to minimize the distance of two positive words. Let $\mathcal{L}^*_{\sim}$ be a set of word index pairs (v, w) for which $l^*(v) = l^*(w)$ holds. In contrast to Equation 7.2.3, we now want to minimize the overall distance. Thus, the objective is given by:

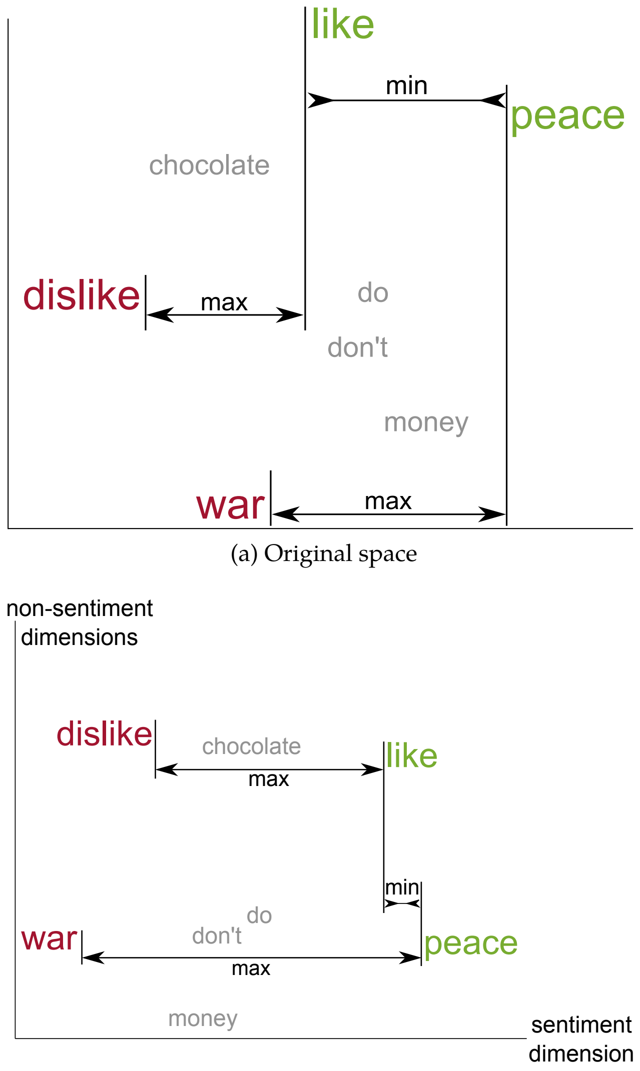
$$\underset{Q}{\operatorname{argmin}} \sum_{(v,w)\in\mathcal{L}^*_{\sim}} \|P^*Q(e_w - e_v)\| \tag{7.2.5}$$

subject to Q being an orthogonal matrix.

The intuition behind the two objectives is graphically depicted in Figure 7.2.1.

7.2.3 *Training*

We combine the two objectives in Equation 7.2.3 and Equation 7.2.5 for each subspace, i.e., for sentiment, concreteness, and frequency, and weight them with α^* and $1 - \alpha^*$. Hence, there is one hyperparameter α^* for each subspace. We then perform Stochastic Gradient Descent (SGD). The batch-size is 100 and the starting learning rate is 5. It is multiplied by 0.99 in each iteration.



(a) Original space

(b) Transformed space

FIGURE 7.2.1: ORIGINAL AND TRANSFORMED SPACE The original word embedding space E^o (top) and the transformed embedding space E^u (bottom). The training objective for Q is to *minimize* the distances in the sentiment dimension between words of the same group (e.g., positive / green: "like" & "peace") and to *maximize* the distances between words of different groups (e.g., negative / red & positive / green: "war" & "peace". The words do not necessarily need to be antonyms).

7.2.4 *Orthogonalization*

Each step of SGD updates Q. The updated matrix Q' is in general no longer orthogonal. We therefore reorthogonalize Q' in each step based on Singular Value Decomposition (SVD):

$$Q' = USV^T \tag{7.2.6}$$

where S is a diagonal matrix, and U and V are orthogonal matrices. The matrix

$$Q := UV^T \tag{7.2.7}$$

is the nearest orthogonal matrix to Q' in both the 2-norm and the Frobenius norm (Fan and Hoffman, 1955). (Formalizing our regularization directly as projected gradient descent would be desirable. However, gradient descent includes an additive operation and orthogonal matrices are not closed under summation.)

SGD for this problem is sensitive to the learning rate. If the learning rate is too large, a large jump results and the reorthogonalized matrix Q basically is a random new point in the parameter space. If the learning rate is too small, then learning can take long. We found that our training regime of starting at a high learning rate (5) and multiplying by 0.99 in every iteration is effective. Typically, the cost initially stays approximately constant (random jumps in parameter space), then cost steeply declines in a small number of about 50 iterations (sweet spot); the curve flattens after that. Training Q took less than 5 minutes per experiment for all experiments in this chapter.

7.3 LEXICON CREATION

For lexicon creation, the input is a set of embeddings and a lexicon resource l, in which words are annotated for a lexical information such as sentiment, concreteness, or frequency. DENSIFIER is then trained to produce a one-dimensional ultradense subspace. The output is an output lexicon. It consists of all words covered by the embedding set, each associated with its one-dimensional ultradense subspace representation (which is simply a real number), an indicator of the word's strength for that information.

The embeddings and lexicon resources used in this chapter cover three lexical properties (sentiment, concreteness, frequency), five languages (Czech (cz), English (en), French (fr), German (de), Spanish (es)), and three domains (news, Twitter, web). Table 7.3.1 lists statistics about the embeddings training corpora for all languages and domains. The Google News embeddings for English[1] and the FrWac embeddings for French[2] are publicly available. We use word2vec to train

[1] https://code.google.com/p/word2vec/

[2] http://fauconnier.github.io/

LANG.	DOMAIN	NAME	# TOKENS	# TYPES
cz	web	CWC2011	3.3B	2.4M
de	web	COW14	11.9B	1.3M
en	news	Google News	100.0B	3.0M
	Twitter	custom	5.4B	3.3M
es	web	COW14	3.7B	0.4M
fr	web	FrWac	1.6B	0.1M

TABLE 7.3.1: EMBEDDINGS TRAINING CORPORA List of embeddings training corpora with basic information, reference, and size. # *tokens*: number of tokens in the corpus. # *types*: number of word types we train embeddings for.

400-dimensional embeddings for English on a custom Twitter corpus of size $3.3e^{12}$ that was collected in 2013. For Czech we use the CWC2011 corpus with a size of $2.4e^{12}$ tokens (Spoustová and Spousta, 2014). For German and Spanish, we train embeddings on web data from the COW project (Schäfer and Bildhauer, 2012; Schäfer, 2015), having sizes of $1.3e^{12}$ and $0.4e^{12}$ tokens, respectively.

We use the following lexicon resources for sentiment (cf. Table 7.3.2): SubLex 1.0 (Veselovská and Bojar, 2013) for Czech; WHM for English (the combination of MPQA (Wilson et al., 2005), Opinion Lexicon (Hu and Liu, 2004), and NRC Emotion lexicons (Mohammad and Turney, 2013)); FEEL (Abdaoui et al., 2014) for French; German Polarity Clues (Waltinger, 2010) for German; and the sentiment lexicon of Pérez-Rosas et al., (2012) for Spanish. For concreteness, we use BWK, a lexicon of 40K English words (Brysbaert et al., 2014). For frequency, we exploit the fact that word2vec stores words in frequency order. Thus, the ranking provided by word2vec is our lexicon resource for frequency.[3]

For a resource / embedding-set pair (l, E), we intersect the vocabulary of l with the top 80K words of E to filter out noisy, infrequent words, because they tend to have low quality embeddings and we do not want them to introduce noise when training the transformation matrix.

For the sentiment and concreteness resources, $l^*(w) \in \{-1, 1\}$ for all words w covered. We create a resource l^f for frequency by setting $l^f(w) = 1$ for the 2K most frequent words and $l^f(w) = -1$ for words at ranks 20K-22K. 1K words randomly selected from the 5K most frequent are the test set.[4] We designate three sets of dimensions D^s, D^c, and D^f

[3] We cannot directly use the token frequency, because token counts are not available for Google News and FrWac embeddings.

[4] The main result of the frequency experiment below is that Kendall's τ is low even in a setup that is optimistic due to train / test overlap; presumably it would be even lower without overlap.

				TRAIN			TEST			
	PROPERTY	LANG.	DOMAIN	RESOURCE	∩	# WORDS	RESOURCE	∩	# WORDS	τ
1	sentiment	cz	web	SubLex 1.0	2,492	4,125	SubLex 1.0	319	500	.580
2	sentiment	de	web	German PC	10,718	37,901	German PC	573	1,000	.654
3	sentiment	es	web	full-strength	824	1,147	full-strength	185	200	.563
4	sentiment	fr	web	FEEL	7,496	10,979	FEEL	715	1,000	.544
5	sentiment	en	Twitter	WHM all	12,601	19,329	Trial 10E	198	200	.661
6	sentiment	en	news	WHM train	7,633	10,270	WHM val	952	1,000	.622
7	concreteness	en	news	BWK	14,361	29,954	BWK	8,694	10,000	.623
8	frequency	en	news	word2vec order	4,000	4,000	word2vec order	1,000	1,000	.361
9	frequency	fr	web	word2vec order	4,000	4,000	word2vec order	1,000	1,000	.460

TABLE 7.3.2: ULTRADENSE LEXICONS Results of lexicon creation for three lexical properties, five languages, and three domains. For each resource, we give its size ("# words") and the size of the intersection of resource and embedding set ("∩"). Kendall's τ is computed on the intersection "∩".

EN-TWITTER		EN-NEWS	
POSITIVE	NEGATIVE	POSITIVE	NEGATIVE
#blessed	rape	expertise	angry
inspiration	racist	delighted	delays
blessed	horrible	honored	worse
inspiring	nasty	thank	anger
foundation	jealousy	wonderful	foul
provide	murder	commitment	blamed
wishes	waste	affordable	blame
dedicated	mess	passion	complained
offers	disgusting	exciting	bad
#happy	spam	flexibility	deaths

TABLE 7.4.1: TOP 10 ENGLISH SENTIMENT WORDS Top 10 sentiment words in the output lexicons for the English Twitter and news domains.

to represent sentiment, concreteness and frequency, respectively, and arbitrarily set (i) $D^c := \{11\}$ for English and $D^c := \emptyset$ for the other languages, because we do not have concreteness resources for them, (ii) $D^s := \{1\}$, and (iii) $D^f := \{21\}$. Referring to the lines in Table 7.3.2, we then learn six orthogonal transformation matrices Q: for cz-web (1), de-web (2), es-web (3), fr-web (4, 9), en-Twitter, and (5) en-news (6, 7, 8).

7.4 EVALUATION

7.4.1 *Top-Ranked Words*

Table 7.4.1 shows the top 10 positive / negative words (i.e., the most extreme values on dimension D^s) when we apply the transformation to the corpora en-Twitter and en-news. Table 7.4.2 shows the top 10 positive / negative words of de-web and the top 10 concrete / abstract words (i.e., most extreme values on dimension D^c) for en-news. For en-Twitter (leftmost double column in Table 7.4.1), the selected words look promising: they contain highly domain-specific words such as hashtags (e.g., #happy). This is surprising because there is not a single hashtag in the lexicon resource WHM that DENSIFIER was trained on. Results for the other double column show likewise extreme examples for the corresponding information and language. This initial evaluation indicates that our method effectively learns high quality lexicons for new domains.

EN-NEWS		DE-WEB	
CONCRETE	ABSTRACT	POSITIVE	NEGATIVE
tree	fundamental	herzlichen	gesperrt
truck	obvious	kenntnisse	droht
kitchen	legitimate	hervorragende	verurteilt
dog	reasonable	ideale	gefahr
bike	optimistic	bestens	falsche
bat	satisfied	glückwunsch	streit
garden	surprising	optimale	angst
homer	honest	anregungen	krankheit
bed	regard	freuen	falschen
gallon	extraordinary	kompetenzen	verdacht

TABLE 7.4.2: TOP 10 ENGLISH AND GERMAN WORDS IN DIFFERENT CATEGORIES Top 10 words in the output lexicons for English concreteness in the news domain and German sentiment in the web domain.

Figure 7.4.1 depicts values for selected words for the three properties. Illustrative examples are "brother" / "brotherhood" for concreteness and "hate" / "love" for sentiment.

7.4.2 *Quality of Predictions*

Table 7.3.2 presents our experimental results. In each case, we split the resource into training and test sets, except for Twitter where we use the trial data of SemEval 2015 Task 10E for test. We train DENSIFIER on the training set and compute Kendall's τ on the test set.

The size of the lexicon resource has no big effect. For example, results for Spanish (small resource; line 3 in Table 7.3.2) and French (large resource; line 4) are about the same. See Section 7.5.2 for a more detailed analysis of the effect of resource size.

The quality of the output lexicon depends strongly on the quality of the underlying word embeddings. For instance, results for French (small embedding training corpus; line 4 in Table 7.3.2) are worse than results for English (large embedding training corpus; line 6) even though the lexicon resources have comparable size. However, the difference may also be caused by the used training resources, because they are from a non-web domain, which might give the English news data an advantage over the French web data. Future work needs to assess on this issue.

In contrast to sentiment and concreteness, τ values for frequency are low (lines 8-9 in Table 7.3.2). For the other three languages we obtain $\tau \in [.34, .46]$ for frequency (not shown). This suggests that word

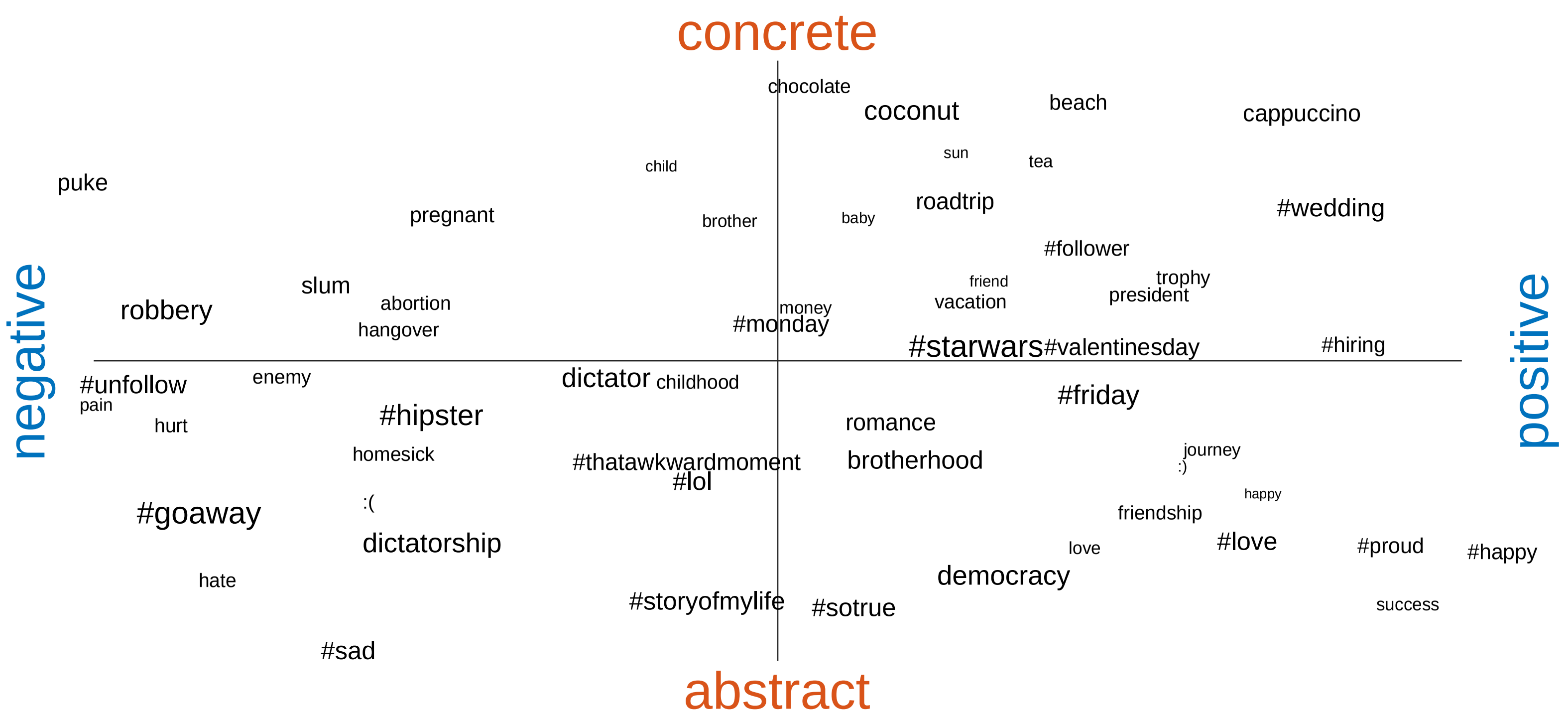


FIGURE 7.4.1: ILLUSTRATION OF EN-TWITTER OUTPUT LEXICON DENSIFIER values are x coordinate (sentiment), y coordinate (concreteness), and font size (frequency).

embeddings represent sentiment and concreteness much better than frequency. The reason for this likely is the learning objective of word embeddings, namely modeling the context. Infrequent words can occur in frequent contexts. Thus, the frequency information in a single word embedding is limited. In contrast negative words are likely to occur in negative contexts.

The nine output lexicons in Table 7.3.2 – each a list of words annotated with predicted strength on one of three properties – are available at www.cis.lmu.de/~sascha/Ultradense/.

7.4.3 *Determining Association Strength*

We also evaluate lexicon creation on SemEval 2015 Task 10E (Rosenthal et al., 2015). As before, the task is to predict the sentiment score of words and phrases. We use the trial data (200 examples) of the task to tune the hyperparameter, $\alpha^s = 0.4$. Out-of-Vocabulary (OOV) words are predicted as neutral (7/1315). Table 7.4.3 shows that the lexicon computed by DENSIFIER (line 5 in Table 7.3.2) has a τ of 0.654 (line 6, column *all* in Table 7.4.3), significantly better than all other systems, including the winner of SemEval 2015 ($\tau = 0.626$, line 1). DENSIFIER also beats Sentiment140 (Mohammad et al., 2013), a widely used semi-automatic sentiment lexicon.

The last column shows Kendall's τ on the intersection of DENSIFIER and Sentiment140. It shows that DENSIFIER again performs significantly better than Sentiment140.

7.4.4 *Polarity Classification*

After describing the formalism of DENSIFIER and showing the quality of the resulting lexicons, we now turn back to polarity classification as extrinsic evaluation. More precisely, we show that ultradense embeddings decrease model training times without any noticeable decrease in performance compared to the original embeddings. We again evaluate on SemEval 2015 Task 10B, classification of Twitter tweets as positive, negative, or neutral (Table 6.3.10 gives dataset statistics). As classification model we use the linguistically-informed Convolutional Neural Network (LINGCNN) (cf. Chapter 5). We do not use sentence-based features to focus on the evaluation of the embeddings. We initialize the first layer of LINGCNN, the embedding layer, in three different ways:

1. 400-dimensional Twitter embeddings (Section 7.3)
2. 40-dimensional ultradense embeddings derived from (i)
3. 4-dimensional ultradense embeddings derived from (i).

The objective weighting is $\alpha^s = .4$, optimized on the development set (cf. Table 6.3.10).

		τ	
	SYSTEM	ALL	$\cap$
1	Amir et al., (2015)	$0.626^\dagger$	
2	Hamdan et al., (2015)	$0.621^\dagger$	
3	Zhang et al., (2015)	$0.591^\dagger$	
4	Özdemir and Bergler, (2015)	$0.584^\dagger$	
5	Plotnikova et al., (2015)	$0.577^\dagger$	
6	Densifier	**0.654**	**0.650**
7	Sentiment140	$0.508^\dagger$	$0.538^\dagger$
8	Densifier, trial only	$0.627^\dagger$	

TABLE 7.4.3: RESULTS OF ASSOCIATION STRENGTH The first "τ" column gives the correlation with the entire test lexicon of SemEval 2015 10E, the last column only on the intersection of our output lexicon and Sentiment140. Of the 1315 words of task 10E, 985 and 1308 are covered by Densifier and Sentiment140, respectively. Significance (Fisher z-transformation) is compared to the best system in the same column with †: $p = 0.05$. Bold is the best performance per column.

We choose the following hyperparameters: filters spanning 2-5 words (100 filters each), k-max pooling with $k = 1$, training with SGD using AdaGrad (Duchi et al., 2011), ℓ_2 regularization ($\lambda = 5e^{-5}$), learning rate of $lr = 0.01$, and mini-batch size of 100.

As before we report macro F_1 of positive and negative classes (the official SemEval evaluation metric) and accuracy over the three classes. Table 7.4.4 shows that 40-dimensional ultradense embeddings perform almost as well as the full 400-dimensional embeddings. There is no significant difference according to a sign test. Training time is shorter by a factor of 21 (85/4 examples/second). The 4-dimensional ultradense embeddings lead to only a small loss of 1.5% although the size of the embeddings is smaller by a factor of 100 (again not a significant drop). The training time is shorter by a factor of 44 (178/4).

We perform the same experiment on CSFD (see Table 6.3.10) to show the benefits of ultradense embeddings for a low-resource language where only one rather small lexicon is available. As original word embeddings we train new 400 dimensional embeddings on a large Twitter corpus ($3.3e^9$ tokens). We use Densifier to create 40 and 4 dimensional embeddings out of these embeddings and SubLex 1.0 (Veselovská and Bojar, 2013). We use the same word-level features as before (see Section 6.3.3). Since CSFD is a large dataset, we randomly split the 91K dataset instances into 90% training and 10% test and report accuracy and macro F_1 score over all three classes.

LANG.	EMBEDDINGS	# DIM	ACC.	F_1	EX./SEC
EN	original	400	66.61	62.35	4
	DENSIFIER	40	66.23	62.02	85
	DENSIFIER	4	64.60	60.76	178
CZ	original	400	80.30	80.21	1
	DENSIFIER	40	80.30	80.10	24
	DENSIFIER	4	77.10	76.90	83

TABLE 7.4.4: POLARITY CLASSIFICATION RESULTS Accuracy and macro F_1 performance of LINGCNN for different embeddings settings. For English, macro F_1 is computed for positive and negative classes. For Czech, it is computed on all three classes.

Table 7.4.4 confirms the findings on English. There is only a small performance drop when using ultradense embeddings (not significant for 40 dimensional embeddings) while the speed improvement is substantial.

7.5 PARAMETER ANALYSIS

In this section, we analyze the influence of two parameters on the quality of ultradense embeddings: (i) the size of ultradense subspace and (ii) the size of lexicon resource. We leave an evaluation of another parameter, the size of the embedding training corpus, for future work, but empirical results suggest that this corpus should ideally have a size of several billion tokens.

7.5.1 *Size of Subspace*

With the exception of the two polarity classification experiments, all our subspaces have dimensionality $d^* = 1$. The question arises: does a one-dimensional space perhaps have too low a capacity to encode all relevant information and could we further improve our results by increasing the dimensionality of the subspace to values $d^* > 1$? The lexicon resources that we train and test on are all binary. Thus, if we use values $d^* > 1$, then we need to map the subspace embeddings to a one-dimensional scale for evaluation. We do this by training, on the train part of the resource, a linear transformation from the ultradense subspace to the one-dimensional scale (e.g., to the sentiment scale).

Figure 7.5.1 compares different values of d^s for three different types of subspaces in this setup, i.e., the setup in which the subspace representations are mapped via linear transformation to a one-dimensional sentiment value:

RANDOM We take the first d^s dimensions of the original embeddings.

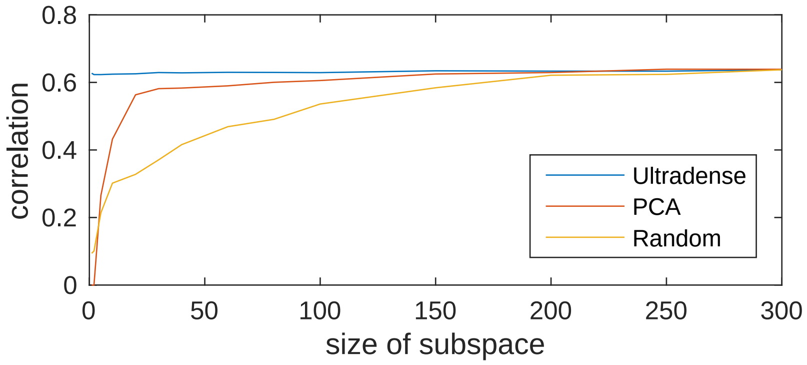


FIGURE 7.5.1: SUBSPACE SIZE ANALYSIS Kendall's τ for different subspace sizes. See line 6 in Table 7.3.2 for training and test split.

PCA We compute a Principal Component Analysis (PCA) and take the first d^s principal components, i.e., those dimensions that comprise the largest variance.

ULTRADENSE We use the ultradense subspace of dimensionality d^s.

We use the word embeddings and lexicon resources of line 6 in Table 7.3.2. For random, the performance starts dropping when the subspace is smaller than 200 dimensions. For PCA, the performance is relatively stable until the subspace becomes smaller than 100 dimensions. In contrast, ultradense subspaces have almost identical performance for all values of d^s, even for $d^s = 1$. This suggests that a single dimension is sufficient to encode all sentiment information needed for sentiment lexicon creation. However, for other sentiment tasks more dimensions may be needed, e.g., for modeling different emotional dimensions of polarity: fear, sadness, anger etc.

An alternative approach to create a low-dimensional space is to simply train low-dimensional word2vec embeddings. The following experiment suggests that this does not work very well. We used word2vec to train 60-dimensional Twitter embeddings with the same settings as on line 5 in Table 7.3.2. While the correlation for 400-dimensional embeddings shown in Table 7.3.2 is 0.661, the correlation of 60-dimensional embeddings is only 0.568. Thus, although we show that the information in 400-dimensional embeddings that is relevant for sentiment can be condensed into a single dimension, hundreds of dimensions seem to be needed if we use word2vec to collect sentiment information. If we run word2vec with a small dimensionality, only a subset of available sentiment information is "harvested" from the corpus.

7.5.2 *Size of Training Resource*

Next, we analyze what size of training resource is required to learn a good transformation Q. Labeled resources covering many words may

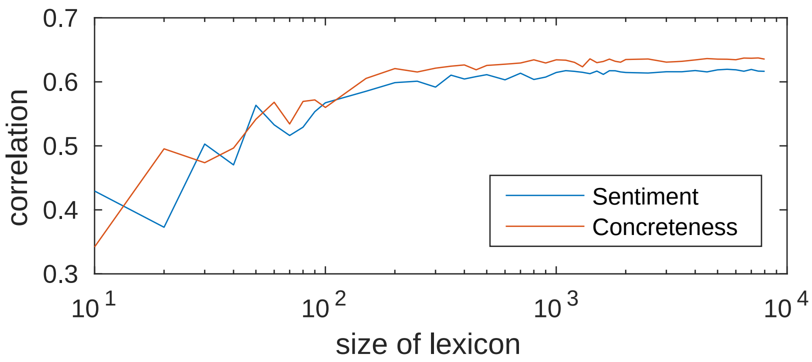


FIGURE 7.5.2: LEXICON SIZE ANALYSIS Kendall's τ for different training resource sizes. See line 8 in Table 7.3.2, for training and test split.

not be available or suffer from lack of quality. This is for example true for less studied languages. We use the settings of lines 6 (sentiment) and 7 (concreteness) in Table 7.3.2. Figure 7.5.2 shows that a small training resource of 300 entries is sufficient for high performance. This suggests that DENSIFIER can create a high quality output lexicon for a new language by hand-labeling only 300 words; and that a small, high-quality resource may be preferable to a large lower-quality resource (semi-automatic or out of domain).

To provide further evidence for this we repeat the association strength experiment from Section 7.4.3. This time however, we train DENSIFIER on only the trial data of SemEval 2015 task 10E, instead of the WHM lexicon. To convert the continuous trial data to binary -1 / 1 labels, we discard all words with sentiment values between -0.5 and 0.5 and round the remaining values, giving us 39 positive and 38 negative training words. We tune α^s on the train set, which in this setting is equal to the trial data of SemEval 2015 task 10E. This seems to work due to the different objectives for training (maximize / minimize difference) and development (correlation).

The resulting lexicon reaches $\tau = 0.627$ (see line 8 in Table 7.4.3). This is worse than $\tau = 0.654$ (line 6) for the setup in which we used several large resources. However, our system would still reach the best rank in the SemEval 2015 Task 10E competition, with only 77 training examples. This indicates that DENSIFIER is especially suited for languages or domains for which little training data is available.

7.6 RELATED WORK

To the best of our knowledge, the presented approach is the first to train an orthogonal transformation to reorder word embedding dimensions into ultradense subspaces. However, there is much prior work on post-processing word embeddings.

Faruqui et al., (2015) perform postprocessing based on a semantic lexicon with the goal of fine-tuning word embeddings. Their transformation is not orthogonal and therefore does not preserve distances. They show that their approach optimizes word embeddings for a given application, i.e., word similarity, but also that it worsens them for other applications like detecting syntactic relations. Faruqui et al., (2015)'s approach also does not have the benefit of ultradense embeddings, in particular the benefit of increased efficiency.

In a tensor framework, Rothe and Schütze, (2015) transform the word embeddings to sense (synset) embeddings. In their work, all embeddings live in the same space whereas we explicitly want to change the embedding space to create ultradense embeddings with several desirable properties.

Xing et al., (2015) restrict the work of Mikolov, Le, et al., (2013) to an orthogonal transformation to ensure that normalized embeddings stay normalized. This transformation is learned between two embedding spaces of different languages to exploit similarities. They normalize word embeddings in a first step, something that does not improve our results.

As a reviewer pointed out, our method is also related to Oriented PCA (Diamantaras and Kung, 1996). However in contrast to PCA a solution for Oriented PCA is not orthogonal.

Sentiment lexicons are often created semi-automatically, e.g., by extending manually labeled seed sets of sentiment words or adding for each word its synonyms and antonyms. Alternatively, words frequently cooccurring with a seed set of manually labeled sentiment words are added (Kiritchenko et al., 2014; Turney, 2002). Heerschop et al., (2011) use WordNet together with a PageRank-based algorithm to propagate the sentiment of the seed set to unknown words. Scheible, (2010) present a semi-automatic approach based on machine translation of sentiment lexicons. The winning system of SemEval 2015 10E (Amir et al., 2015) is based on structured skip-gram embeddings with 600 dimensions and support vector regression with RBF kernels. Hamdan et al., (2015), the second ranked team, use the average of six sentiment lexicons as a final sentiment score, a method that cannot be applied to low resource languages. We show that the lexicons created by DENSIFIER achieve better performance than other semi-automatically created lexicons.

Tang, Wei, Yang, et al., (2014) train sentiment specific embeddings by extending Collobert and Weston, (2008)'s model and Tang, Wei, Qin, Zhou, et al., (2014)'s skip-gram model. The first model automatically labels tweets as positive and negative based on emoticons, a process that cannot be easily transferred to other domains like news. The second uses the Urban Dictionary to expand a small list of 350 sentiment seeds. In our work, we show that a training resource of about the same size is sufficient without an additional dictionary. DENSIFIER differs from this work in that it does not need a text corpus, but can transform existing, publicly available word embeddings. DENSIFIER is independent of the embedding learning algorithm and therefore extensible to other word embedding models like GloVe (Pennington et al., 2014), to phrase embeddings (Yu and Dredze, 2015), and even to sentence embeddings (Kiros et al., 2015).

7.7 CONCLUSION

We described DENSIFIER, a method that transforms task-agnostic word embeddings to an ultradense subspace that contains only the information relevant for the application. In experiments on SemEval, DENSIFIER demonstrates two benefits of the ultradense subspace. (i) Information is preserved even if we focus on a subspace that is smaller by a factor of 100 than the original space. This means that unnecessary noisy information is removed from the embeddings and robust learning without overfitting is better supported. (ii) Since the subspace is 100 times smaller, models that use the embeddings as their input representation can be trained more efficiently and have a much smaller number of parameters. We could speed up the classifier training by a factor of 44.

The subspace can be learned with just 80-300 training examples, achieving state-of-the-art results on lexicon creation. The nine large DENSIFIER lexicons shown in Table 7.3.2 are publicly available.[5]

We have described in this chapter that up to three orthogonal ultradense subspaces can be created. Many training datasets can be restructured as sets of similar and dissimilar pairs. For instance, in part-of-speech tasks verb/verb pairs would be similar, verb/noun pairs dissimilar. Hence, our objective is widely applicable. Therefore, we propose to explore the possibility of factoring all information present in an embedding into a dozen or so orthogonal subspaces. This factorization would not change the information embeddings contain, but it would make them more compact for any given application, more meaningful and more interpretable.

7.8 FUTURE WORK

We propose the following extensions to the presented work:

- In Section 7.4.2 we show that the quality of word embeddings determines the quality of the output lexicons. More research has to analyze influences of the embeddings training corpus, for example its optimal size. Moreover, the interaction of the embeddings corpus' domain with the domain of the lexicon resource must be analyzed. For example, can DENSIFIER successfully be

[5] www.cis.lmu.de/~sascha/Ultradense/

trained with news-based word embeddings and a Twitter lexicon? The qualitative analysis in Table 7.4.1 suggests that, but a more profound and quantitative experiment could give clarity on this matter.

- Our polarity classification experiments in Section 7.4.4 suggest that there is a minimum number of embeddings dimensions for LINGCNN where no drop in performance is created and still the training time is much smaller. This optimal point can be found by more experimentation.
- When creating a multi-dimensional sentiment space for polarity classification – having 40 or 4 dimensions – the actual meaning of every single dimension is still unknown. One possible future direction would be to analyze each dimension on its own. Possible information that may be encoded are negation or valence values for different word senses, although the new dimensions are likely to be just as distributed as the original dimensions.
- Since only 300 labeled words in the lexicon resource are required to create a high-quality sentiment lexicon, new lexicons for many languages can relatively easy be created for low-resource languages. Using Wikipedia as embeddings training corpus and 300 hand-labeled words would allow to easily provide a large number of lexicons at small cost.

8 CONCLUSION

As we saw, Sentiment Analysis (SA) is a challenging research area, which faces many difficulties, such as lack of resources, and requires semantic understanding to capture all nuances of polarity. In this thesis we addressed some of the problems.

1. We have addressed the issue of sense-dependent polarity by proposing the idea of a Contextually Enhanced Sentiment Lexicon (CESL). Our approach of analyzing the senses of a word in the light of sentiment showed that there exist sentiment-specific differences in the meaning of words. A detailed and complete analysis of a text is only possible by having a fine-grained understanding of sentiment-related word senses. The presented approach however is labeling intense and therefore requires more research to find alternative labeling strategies, along the lines of the presented semi-automatic clustering approach.

2. We have extended an existing Convolutional Neural Network (CNN) architecture with linguistic knowledge. Since SA is a very semantic topic, statistical models can be supported and enhanced by existing resources such as sentiment lexicons. These resources alone however cannot be used for a well working polarity classification system, because they contain only *prior* polarity labels. In other words, they are context independent. Thus, prior knowledge together with a statistical model, which considers a word's context, lead to powerful systems. This is not contradictory to our claim that a detailed and complete analysis of the text requires fine-grained sentiment-related word senses. First, linguistically-informed Convolutional Neural Network (lingCNN) as presented classifies polarity of the entire sentence and neither gives an explanation nor a detailed analysis why a sentence is positive or negative. Second, if such an analysis is requested, lingCNN can be used to classify the sense of a word given its context. By that it can benefit from the CESL.

3. We have shown that discarding morphological information does not harm polarity classification. It is even beneficial. This is true although intuitively morphology seems important. For instance a lemmatizer maps "good", "better", "best" to the same lemma "good". We claim that for polarity classification comparatives and superlatives are not as important. Some inflection, such as the number and gender of nouns and adjectives, and conjugation of verbs, such as gender, tense, etc, do not seem to add any senti-

ment information. But for valence prediction the normalization may hurt performance. For instance, the mood of a word (e.g., subjunctive vs. imperative) is able to change the magnitude of sentiment. This is true for English. For other languages, especially Morphologically Rich Languages (MRLs), different rules may apply. Further research is necessary to fully understand the influence of morphology on SA.

4. We have presented three different approaches to address sparsity issues that emerge out of a lack of data. The non-linear extension to the Log-Bilinear Language model (LBL) model has proven to be helpful especially when the model has only a small number of parameters, which usually is the case when little training data is available. The use of linguistic knowledge in the lingCNN has been beneficial when little training data is available. And finally, the normalization of corpora before training a polarity classification system has proven beneficial for languages with a rich morphology, because it reduces the sparsity related to rare or missing word forms.

5. We have described a method that is able to create large-scale lexicons (e.g., sentiment lexicons) for low-resource languages by requiring a minimum of manual labeling effort. We have shown that this method increases efficiency of models that use word embeddings for specific tasks, such as polarity classification. By reducing the number of dimensions from 400 down to 4 and compressing all sentiment information into an ultradense subspace, without lowering the performance much we can train bigger models or train models on more data in less time.

The ultimate goal of SA should be the creation of user-specific models. Every user has a different view about polarity, has different understanding of, or feelings about sarcasm and irony. A model that is dedicated to a single user's preference can assist in finding interesting articles to read, movies to watch, etc. In such a scenario data scarcity is one of the biggest problems. The user had to label every text (e.g., news article, movie/book description) by how much he likes it, i.e., what his sentiment is towards that text. The methods presented in this thesis can help achieving this goal by reducing the need for data and by providing model choices that help in the final classification.

ACRONYMS

ASR	Automatic Speech Recognition
BOW	Bag-of-Words
CBOW	Continuous Bag-of-Words
CESL	Contextually Enhanced Sentiment Lexicon
CNN	Convolutional Neural Network
CSFD	Czech Film Database
CSLM	Continuous Space Language Model
KN	Kneser-Ney
Lamb	LemmA eMBeddings
LBL	Log-Bilinear Language model
lingCNN	linguistically-informed Convolutional Neural Network
LM	Language Model
LSTM	Long Short Term Memory
MLE	Maximum Likelihood Estimate
MLP	Multi Layer Perceptron
ML	Machine Learning
MRL	Morphologically Rich Language
MRR	Mean Reciprocal Rank
MT	Machine Translation
NCE	Noise-Contrastive Estimation
NER	Named Entity Recognition
nLBL	non-linear Log-Bilinear Language model
NLP	Natural Language Processing
NNLM	Neural Network Language Model
NN	Neural Network
nvLBL	non-linear vectorized Log-Bilinear Language model

OCR	Optical Character Recognition
OOV	Out-of-Vocabulary
PCA	Principal Component Analysis
PCD	Predicted Context Distribution
POS	Part-of-Speech
PPL	Perplexity
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SA	Sentiment Analysis
SGD	Stochastic Gradient Descent
Stem	STem EMbeddings
SVD	Singular Value Decomposition
SVM	Support Vector Machine
vLBL	vectorized Log-Bilinear Language model
WSD	Word Sense Disambiguation
WSJ	Wall Street Journal

BIBLIOGRAPHY

Amine Abdaoui, Jérôme Azé, Sandra Bringay, and Pascal Poncelet (2014). *FEEL: French Extended Emotional Lexicon: ISLRN: 041-639-484-224-2*.

Cem Akkaya, Janyce M. Wiebe, and Rada Mihalcea (2009). "Subjectivity Word Sense Disambiguation." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Vol. 1.

Silvio Amir, Ramón Astudillo, Wang Ling, Bruno Martins, Mario J. Silva, and Isabel Trancoso (2015). "INESC-ID: A Regression Model for Large Scale Twitter Sentiment Lexicon Induction." In: *Proceedings of the 9th International Workshop on Semantic Evaluation*.

Ebru Arisoy, Tara N. Sainath, Brian Kingsbury, and Bhuvana Ramabhadran (2012). "Deep Neural Network Language Models." In: *Proceedings of the NAACL-HLT Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani (2010). "SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining." In: *Proceedings of the International Conference on Language Resources and Evaluation*.

Marco Baroni and Sabrina Bisi (2004). "Using Cooccurrence Statistics and the Web to Discover Synonyms in a Technical Language." In: *Proceedings of the Fourth International Conference on Language Resources and Evaluation*.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski (2014). "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors." In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent (2000). "A Neural Probabilistic Language Model." In: *Proceedings of the Advances in Neural Information Processing Systems 13*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin (2003). "A Neural Probabilistic Language Model." In: *Journal of Machine Learning Research* 3, pp. 1137–1155.

Yoshua Bengio (2009). "Learning Deep Architectures for AI." In: *Foundations and Trends in Machine Learning* 2.1, pp. 1–127.

William Blacoe and Mirella Lapata (2012). "A Comparison of Vector-based Representations for Semantic Composition." In: *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Francis Bond and Kyonghee Paik (2012). "A Survey of Wordnets and their Licenses." In: *Proceedings of the 6th Global WordNet Conference*.

Jan A. Botha and Phil Blunsom (2014). "Compositional Morphology for Word Representations and Language Modelling." In: *Proceedings of the 31st International Conference on Machine Learning*.

Margaret M. Bradley and Peter J. Lang (1999). *Affective norms for English words (ANEW): Instruction manual and affective ratings*.

Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souza, Jennifer C. Lai, and Robert L. Mercer (1992). "Class-Based n-gram Models of Natural Language." In: *Computational Linguistics* 18.4, pp. 467–479.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni (2014). "Multimodal Distributional Semantics." In: *Journal of Artificial Intelligence Research* 49, pp. 1–47.

Tomas Brychcin and Ivan Habernal (2013). "Unsupervised Improving of Sentiment Analysis Using Global Target Context." In: *Recent Advances in Natural Language Processing*.

Marc Brysbaert, Amy B. Warriner, and Victor Kuperman (2014). "Concreteness ratings for 40 thousand generally known English word lemmas." In: *Behavior Research Methods* 46.3, pp. 904–911.

John A. Bullinaria and Joseph P. Levy (2007). "Extracting semantic representations from word co-occurrence statistics: A computational study." In: *Behavior Research Methods* 39.3, pp. 510–526.

John A. Bullinaria and Joseph P. Levy (2012). "Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and SVD." In: *Behavior Research Methods* 44.3, pp. 890–907.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn (2013). "One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling." In: *Computing Research Repository* abs/1312.3005.

Stanley F. Chen and Joshua T. Goodman (1999). "An empirical study of smoothing techniques for language modeling." In: *Computer Speech & Language* 13.4, pp. 359–393.

Grzegorz Chrupała (2008). "Towards a Machine-Learning Architecture for Lexical Functional Grammar Parsing." PhD thesis. Dublin City University.

Ronan Collobert and Jason Weston (2008). "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning." In: *Proceedings of the Twenty-Fifth International Conference on Machine Learning*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa (2011). "Natural Language Processing (almost) from Scratch." In: *Journal of Machine Learning Research* 12, pp. 2493–2537.

John S. Denker, W. R. Gardner, Hans Peter Graf, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, Lawrence D. Jackel, Henry S. Baird, and Isabelle Guyon (1988). "Neural Network Recognizer for Hand-Written Zip Code Digits." In: *Proceedings of the Advances in Neural Information Processing Systems 1*.

Konstantinos I. Diamantaras and S. Y. Kung (1996). *Principal Component Neural Networks: Theory and Applications*. Adaptive and Learning Systems for Signal Processing, Communications, and Control Series.

Georgiana Dinu, Nghia The Pham, and Marco Baroni (2013). "General estimation and evaluation of compositional distributional semantic models." In: *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*.

Cícero Nogueira dos Santos and Maíra Gatti (2014). "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts." In: *Proceedings of the 25th International Conference on Computational Linguistics*.

Cícero Nogueira dos Santos and Bianca Zadrozny (2014). "Learning Character-level Representations for Part-of-Speech Tagging." In: *Proceedings of the 31st International Conference on Machine Learning*.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou (2015). "Classifying Relations by Ranking with Convolutional Neural Networks." In: *Proceedings of the 53nd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*.

John C. Duchi, Elad Hazan, and Yoram Singer (2011). "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." In: *Journal of Machine Learning Research* 12, pp. 2121–2159.

Chris Dyer (2013). *Notes on Adagrad*.

Sebastian Ebert and Hinrich Schütze (2014). "Fine-Grained Contextual Predictions for Hard Sentiment Words." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Sebastian Ebert, Ngoc Thang Vu, and Hinrich Schütze (2015a). "A Linguistically Informed Convolutional Neural Network." In: *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*.

Sebastian Ebert, Ngoc Thang Vu, and Hinrich Schütze (2015b). "CIS-positive: Combining Convolutional Neural Networks and SVMs for Sentiment Analysis in Twitter." In: *Proceedings of the 9th International Workshop on Semantic Evaluation*.

Sebastian Ebert, Thomas Müller, and Hinrich Schütze (2016). "LAMB: A Good Shepherd of Morphologically Rich Languages." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*.

Jeffrey L. Elman (1990). "Finding Structure in Time." In: *Cognitive Science* 14.2, pp. 179–211.

Andrea Esuli and Fabrizio Sebastiani (2006). "SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining." In: *Proceedings of the fifth International Conference on Language Resources and Evaluation*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin (2008). "LIBLINEAR: A Library for Large Linear Classification." In: *Journal of Machine Learning Research* 9, pp. 1871–1874.

Ky Fan and Alan J. Hoffman (1955). "Some metric inequalities in the space of matrices." In: *Proceedings of the American Mathematical Society* 6.1, pp. 111–116.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith (2015). "Retrofitting Word Vectors to Semantic Lexicons." In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin (2002). "Placing search in context: the concept revisited." In: *ACM Transactions on Information Systems* 20.1, pp. 116–131.

Michael Gamon (2004). "Sentiment Classification on Customer Feedback Data: Noisy Data, Large Feature Vectors, and the Role of Linguistic Analysis." In: *Proceedings of the 20th International Conference on Computational Linguistics*.

Alec Go, Richa Bhayani, and Lei Huang (2009). *Twitter Sentiment Classification using Distant Supervision*.

Joshua T. Goodman (2001). "A bit of progress in language modeling." In: *Computer Speech & Language* 15.4, pp. 403–434.

Irving J. Good (1953). "The population frequencies of species and the estimation of population parameters." In: *Biometrika* 40.3-4, pp. 237–264.

Gintare Grigonyte, João Cordeiro, Gaël Dias, Rumen Moraliyski, and Pavel Brazdil (2010). "Paraphrase Alignment for Synonym Evidence Discovery." In: *Proceedings of the 23rd International Conference on Computational Linguistics*.

Tobias Günther and Lenz Furrer (2013). "GU-MLT-LT: Sentiment Analysis of Short Messages using Linguistic Features and Stochastic Gradient Descent." In: *Proceedings of the 7th International Workshop on Semantic Evaluation*.

Iryna Gurevych (2005). "Using the Structure of a Conceptual Network in Computing Semantic Relatedness." In: *Proceedings of the Second International Joint Conference on Natural Language Processing*. Lecture Notes in Computer Science (LNCS).

Michael Gutmann and Aapo Hyvärinen (2012). "Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics." In: *Journal of Machine Learning Research* 13, pp. 307–361.

Ivan Habernal, Tomáš Ptáček, and Josef Steinberger (2013). "Sentiment Analysis in Czech Social Media Using Supervised Machine Learning." In: *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*.

Matthias Hagen, Martin Potthast, Michel Büchner, and Benno Stein (2015). "Webis: An Ensemble for Twitter Sentiment Detection." In: *Proceedings of the 9th International Workshop on Semantic Evaluation*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang (2009). "The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages." In: *Proceedings of the 13th Conference on Computational Natural Language Learning: Shared Task*.

Hussam Hamdan, Patrice Bellot, and Frederic Bechet (2015). "Lsislif: Feature Extraction and Label Weighting for Sentiment Analysis in Twitter." In: *Proceedings of the 9th International Workshop on Semantic Evaluation*.

Birgit Hamp and Helmut Feldweg (1997). "GermaNet - a Lexical-Semantic Net for German." In: *In Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*.

Samer Hassan and Rada Mihalcea (2009). "Cross-lingual Semantic Relatedness Using Encyclopedic Knowledge." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Bas Heerschop, Alexander Hogenboom, and Flavius Frasincar (2011). "Sentiment Lexicon Creation from Lexical Resources." In: *Proceedings of the 14th International Conference on Business Information Systems*. Vol. 87. Lecture Notes in Business Information Processing.

Felix Hill, Roi Reichart, and Anna Korhonen (2014). "SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation." In: *Computing Research Repository* abs/1408.3456.

Geoffrey E. Hinton, James L. McClelland, and David E. Rumelhart (1986). "Distributed Representations." In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Ed. by David E. Rumelhart and James L. McClelland. Vol. Vol. 1.

Geoffrey E. Hinton (1984). *Distributed representations*.

Geoffrey E. Hinton (1986). "Learning Distributed Representations of Concepts." In: *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*.

Sepp Hochreiter and H. Jürgen Schmidhuber (1997). "Long Short-Term Memory." In: *Neural Computation* 9.8, pp. 1735–1780.

Minqing Hu and Bing Liu (2004). "Mining and Summarizing Customer Reviews." In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng (2012). "Improving Word Representations via Global Context and Multiple Word Prototypes." In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun (2009). "What is the Best Multi-Stage Architecture for Object Recognition?" In: *Proceedings of the IEEE 12th International Conference on Computer Vision*.

Frederick Jelinek and Robert L. Mercer (1980). "Interpolated Estimation of Markov Source Parameters from Sparse Data." In: *Proceedings of the Workshop on Pattern Recognition in Practice*.

Nitin Jindal and Bing Liu (2008). "Opinion Spam and Analysis." In: *Proceedings of the International Conference on Web Search and Web Data Mining*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom (2014). "A Convolutional Neural Network for Modelling Sentences." In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Jussi Karlgren and Magnus Sahlgren (2001). "From Words to Understanding." In: *Foundations of Real World Intelligence*. Ed. by Yoshinori Uesaka, Pentti Kanerva, and Hideki Asoh.

Slava M. Katz (1987). "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer." In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 35.3, pp. 400–401.

Yoon Kim (2014). "Convolutional Neural Networks for Sentence Classification." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M. Mohammad (2014). "Sentiment Analysis of Short Informal Texts." In: *Journal of Artificial Intelligence Research* 50, pp. 723–762.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler (2015). "Skip-Thought Vectors." In: *Proceedings of the Advances in Neural Information Processing Systems 28*.

Reinhard Kneser and Hermann Ney (1995). "Improved backing-off for M-gram language modeling." In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*.

Maximilian Köper, Christian Scheible, and Sabine Schulte Im Walde (2015). "Multilingual Reliability and "Semantic" Structure of Continuous Word Spaces." In: *Proceedings of the 11th International Conference on Computational Semantics*.

Igor Labutov and Hod Lipson (2013). "Re-embedding Words." In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Thomas K. Landauer and Susan T. Dumais (1997). "A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge." In: *Psychological Review* 104.2, pp. 211–240.

Pierre-Simon Laplace (1825). *Pierre-Simon Laplace Philosophical Essay on Probabilities*. 5th edition, Translated by Andrew I. Dale, 1995.

Hai-Son Le, Alexandre Allauzen, Guillaume Wisniewski, and François Yvon (2010). "Training Continuous Space Language Models: Some Practical Issues." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon (2013). "Structured Output Layer Neural Network Language Models for Speech Recognition." In: *IEEE Transactions on Audio, Speech and Language Processing* 21.1, pp. 197–206.

Quoc V. Le and Tomas Mikolov (2014). "Distributed Representations of Sentences and Documents." In: *Proceedings of the 31st International Conference on Machine Learning*.

Rémi Lebret, Joël Legrand, and Ronan Collobert (2013). "Is Deep Learning Really Necessary for Word Embeddings?" In: *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*.

Yann LeCun, Bernhard E. Boser, John S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1989). "Backpropagation Applied to Handwritten Zip Code Recognition." In: *Neural Computation* 1.4, pp. 541–551.

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel (1990). "Handwritten Digit Recognition with a Back-Propagation." In: *Proceedings of the Advances in Neural Information Processing Systems 2*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-Based Learning Applied to Document Recognition." In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Ira Leviant and Roi Reichart (2015). "Judgment Language Matters: Multilingual Vector Space Models for Judgment Language Aware Lexical Semantics." In: *Computing Research Repository* abs/1508.00106.

Omer Levy, Yoav Goldberg, and Ido Dagan (2015). "Improving Distributional Similarity with Lessons Learned from Word Embeddings." In: *Transactions of the Association for Computational Linguistics* 3, pp. 211–225.

Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso (2015). "Two/Too Simple Adaptations of Word2Vec for Syntax Problems." In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning (2013). "Better Word Representations with Recursive Neural Networks for Morphology." In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*.

Andrew L. Maas and Andrew Y. Ng (2010). "A Probabilistic Model for Semantic Word Vectors." In: *Proceedings of the NIPS Deep Learning and Unsupervised Feature Learning Workshop*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. 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*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze (2009). *Introduction to Information Retrieval*.

Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz (1993). "Building a Large Annotated Corpus of English: The Penn Treebank." In: *Computational Linguistics* 19.2, pp. 313–330.

Andrew J. McMinn, Yashar Moshfeghi, and Joemon M. Jose (2013). "Building a large-scale corpus for evaluating event detection on twitter." In: *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*.

Oren Melamud, Ido Dagan, Jacob Goldberger, Idan Szpektor, and Deniz Yuret (2014). "Probabilistic Modeling of Joint-context in Distributional Similarity." In: *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*.

Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu (2015). "Encoding Source Language with Convolutional Neural Network for Machine Translation." In: *Proceedings of the 53nd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*.

Márton Miháltz, Csaba Hatvani, Judit Kuti, György Szarvas, János Csirik, Gábor Prószéky, and Tamás Váradi (2008). "Methods and Results of the Hungarian WordNet Project." In: *Proceedings of the 4th Global WordNet Conference*.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur (2010). "Recurrent Neural Network Based Language Model." In: *Proceedings of the 11th Annual Conference of the International Speech Communication Association*.

Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukás Burget, and Jan Cernocký (2011). "Empirical Evaluation and Combination of Advanced Language Modeling Techniques." In: *Proceedings of the 12th Annual Conference of the International Speech Communication Association*.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean (2013). "Efficient Estimation of Word Representations in Vector Space." In: *Proceedings of the 1st International Conference on Learning Representations*.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever (2013). "Exploiting Similarities among Languages for Machine Translation." In: *Computing Research Repository* abs/1309.4168.

George A. Miller and Walter G. Charles (1991). "Contextual correlates of semantic similarity." In: *Language and Cognitive Processes* 6.1, pp. 1–28.

George A. Miller (1995). "WordNet: A Lexical Database for English." In: *Communications of the ACM* 38.11, pp. 39–41.

Yasuhide Miura, Shigeyuki Sakaki, Keigo Hattori, and Tomoko Ohkuma (2014). "TeamX: A Sentiment Analyzer with Enhanced Lexicon Mapping and Weighting Scheme for Unbalanced Data." In: *Proceedings of the 8th International Workshop on Semantic Evaluation*.

Andriy Mnih and Geoffrey E. Hinton (2007). "Three New Graphical Models for Statistical Language Modelling." In: *Proceedings of the Twenty-Fourth International Conference on Machine Learning*. Vol. 227. ACM International Conference Proceeding Series.

Andriy Mnih and Geoffrey E. Hinton (2008). "A Scalable Hierarchical Distributed Language Model." In: *Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems*.

Andriy Mnih and Yee Whye Teh (2012). "A fast and simple algorithm for training neural probabilistic language models." In: *Proceedings of the 29th International Conference on Machine Learning*.

Andriy Mnih and Koray Kavukcuoglu (2013). "Learning word embeddings efficiently with noise-contrastive estimation." In: *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu (2013). "NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets." In: *Proceedings of the 7th International Workshop on Semantic Evaluation*.

Saif M. Mohammad and Peter D. Turney (2013). "Crowdsourcing a Word-Emotion Association Lexicon." In: *Computational Intelligence* 29.3, pp. 436–465.

Frederic Morin and Yoshua Bengio (2005). "Hierarchical Probabilistic Neural Network Language Model." In: *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*.

Thomas Müller, Helmut Schmid, and Hinrich Schütze (2013). "Efficient Higher-Order CRFs for Morphological Tagging." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Thomas Müller, Ryan Cotterell, Alexander M. Fraser, and Hinrich Schütze (2015). "Joint Lemmatization and Morphological Tagging with Lemming." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Thomas Müller and Hinrich Schütze (2015). "Robust Morphological Tagging with Word Representations." In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Vinod Nair and Geoffrey E. Hinton (2010). "Rectified Linear Units Improve Restricted Boltzmann Machines." In: *Proceedings of the 27th International Conference on Machine Learning*.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa A. Wilson (2013). "SemEval-2013 Task 2: Sentiment Analysis in Twitter." In: *Proceedings of the 7th International Workshop on Semantic Evaluation*.

Roberto Navigli and Simone Paolo Ponzetto (2012). "BabelRelate! A Joint Multilingual Approach to Computing Semantic Relatedness." In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith (2013). "Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters." In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Canberk Özdemir and Sabine Bergler (2015). "CLaC-SentiPipe: SemEval2015 Subtasks 10 B,E, and Task 11." In: *Proceedings of the 9th International Workshop on Semantic Evaluation*.

Sebastian Padó (2006). *User's guide to sigf: Significance testing by approximate randomisation*.

Karel PALA and Pavel SMRZ (2004). "Building Czech Wordnet." In: *Romanian Journal of Information Science and Technology* 7.1-2.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan (2002). "Thumbs Up?: Sentiment Classification Using Machine Learning Techniques." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Bo Pang and Lillian Lee (2004). "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts." In: *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

Bo Pang and Lillian Lee (2008). *Opinion Mining and Sentiment Analysis*. Vol. 2. Foundations and Trends in Information Retrieval.

Alexandre Passos, Vineet Kumar, and Andrew K. McCallum (2014). "Lexicon Infused Phrase Embeddings for Named Entity Resolution." In: *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Verónica Pérez-Rosas, Carmen Banea, and Rada Mihalcea (2012). "Learning Sentiment Lexicons in Spanish." In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation*.

Rene Pickhardt, Thomas Gottron, Martin Körner, Paul G. Wagner, Till Speicher, and Steffen Staab (2014). "A Generalized Language Model as the Combination of Skipped n-grams and Modified Kneser Ney Smoothing." In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Nataliia Plotnikova, Micha Kohl, Kevin Volkert, Stefan Evert, Andreas Lerner, Natalie Dykes, and Heiko Ermer (2015). "KLUEless: Polarity Classification and Association." In: *Proceedings of the 9th International Workshop on Semantic Evaluation*.

Robert Plutchik (1980). "A general psychoevolutionary theory of emotion." In: *Emotion: Theory, research and experience. Vol. 1, Theories of emotion*. Ed. by Robert Plutchik and Henry Kellerman. Vol. 1.

Livia Polanyi and Annie Zaenen (2004). "Contextual Lexical Valence Shifters." In: *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text Theories and Applications*. Vol. 7.

Livia Polanyi and Annie Zaenen (2006). "Contextual Valence Shifters." In: *Computing Attitude and Affect in Text: Theory and Applications*. Ed. by James G. Shanahan, Yan Qu, and Janyce M. Wiebe. Vol. 20. The Information Retrieval Series.

Thomas Proisl, Paul Greiner, Stefan Evert, and Besim Kabashi (2013). "KLUE: Simple and robust methods for polarity classification." In: *Proceedings of the 7th International Workshop on Semantic Evaluation*.

Marek Rei and Ted Briscoe (2014). "Looking for Hyponyms in Vector Space Language Learning." In: *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*.

Ellen M. Riloff, Janyce M. Wiebe, Michael Collins, and Mark Steedman (2003). "Learning Extraction Patterns for Subjective Expressions." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Ellen M. Riloff, Janyce M. Wiebe, and Theresa A. Wilson (2003). "Learning Subjective Nouns using Extraction Pattern Bootstrapping." In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL*. Vol. 4.

Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov (2014). "SemEval-2014 Task 9: Sentiment Analysis in Twitter." In: *Proceedings of the 8th International Workshop on Semantic Evaluation*.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M. Mohammad, Alan Ritter, and Veselin Stoyanov (2015). "SemEval-2015 Task 10: Sentiment Analysis in Twitter." In: *Proceedings of the 9th International Workshop on Semantic Evaluation*.

Sascha Rothe and Hinrich Schütze (2015). "AutoExtend: Extending Word Embeddings to Embeddings for Synsets and Lexemes." In: *Proceedings of the 53nd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze (2016). "Ultradense Word Embeddings by Orthogonal Transformation." In: *Pro-*

ceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

Herbert Rubenstein and John B. Goodenough (1965). "Contextual correlates of synonymy." In: *Communications of the ACM* 8.10, pp. 627–633.

Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells (2005). "Using context-window overlapping in synonym discovery and ontology extension." In: *Proceedings of the Recent Advances in Natural Language Processing III*.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning representations by back-propagating errors." In: *Letters to Nature* 323.

Magnus Sahlgren (2008). "The distributional hypothesis." In: *Rivista di lingüística* 20.1, pp. 33–54.

Holger Schwenk and Jean-Luc Gauvain (2005). "Training Neural Network Language Models on Very Large Corpora." In: *Proceedings of the Conference on Human Language Technology Conference and Empirical Methods in Natural Language Processing*.

Roland Schäfer and Felix Bildhauer (2012). "Building Large Corpora from the Web Using a New Efficient Tool Chain." In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation*.

Christian Scheible and Hinrich Schütze (2013). "Sentiment Relevance." In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Holger Schwenk (2004). "Efficient Training of Large Neural Networks for Language Modeling." In: *Proceedings of the IEEE International Joint Conference on Neural Networks*.

Holger Schwenk (2007). "Continuous space language models." In: *Computer Speech & Language* 21.3, pp. 492–518.

Christian Scheible (2010). "Sentiment Translation through Lexicon Induction for Computational Linguistics." In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*.

Roland Schäfer (2015). "Processing and querying large web corpora with the COW14 architecture." In: *Proceedings of the 3rd Workshop on Challenges in the Management of Large Corpora (CMLC-3)*.

Hinrich Schütze (1992). "Dimensions of Meaning." In: *Proceedings of Supercomputing*.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie (2013). "Overview of the SPMRL Shared Task: A Cross-Framework Evaluation of Parsing Morphologically

Rich Languages." In: *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Aliaksei Severyn and Alessandro Moschitti (2015). "UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification." In: *Proceedings of the 9th International Workshop on Semantic Evaluation*.

John M. Sinclair (1987). *Looking Up: An account of the COBUILD Project in lexical computing*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning (2011). "Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts (2013). "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Radu Soricut and Franz J. Och (2015). "Unsupervised Morphology Induction Using Word Embeddings." In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Johanka Spoustová and Miroslav Spousta (2014). "A High-Quality Web Corpus of Czech." In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation*.

Philip J. Stone, Dexter C. Dunphy, and Marshall S. Smith (1966). "The General Inquirer: A Computer Approach to Content Analysis." In: *American Educational Research Journal* 4.4, p. 397.

György Szarvas, Torsten Zesch, and Iryna Gurevych (2011). "Combining Heterogeneous Knowledge Resources for Improved Distributional Semantic Models." In: *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing*. Vol. 6608. Lecture Notes in Computer Science.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly D. Voll, and Manfred Stede (2011). "Lexicon-Based Methods for Sentiment Analysis." In: *Computational Linguistics* 37.2, pp. 267–307.

Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou (2014). "Coooolll: A Deep Learning System for Twitter Sentiment Classification." In: *Proceedings of the 8th International Workshop on Semantic Evaluation*.

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu (2014). "Building Large-Scale Twitter-Specific Sentiment Lexicon: A Representation Learning Approach." In: *Proceedings of the 25th International Conference on Computational Linguistics*.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin (2014). "Learning Sentiment-Specific Word Embedding for Twitter

Sentiment Classification." In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer (2015). "Evaluation of Word Vector Representations by Subspace Alignment." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Peter D. Turney, Michael L. Littman, Jeffrey Bigham, and Victor Shnayder (2003). "Combining independent modules in lexical multiple-choice problems." In: *Proceedings of the Recent Advances in Natural Language Processing III*. Vol. 260. Current Issues in Linguistic Theory (CILT).

Peter D. Turney (2001). "Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL." In: *Proceedings of the 12th European Conference on Machine Learning*. Vol. 2167. Lecture Notes in Computer Science.

Peter D. Turney (2002). "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews." In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang (2013). "Decoding with Large-Scale Neural Language Models Improves Translation." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kateřina Veselovská and Ondřej Bojar (2013). *Czech SubLex 1.0*. URL: http://hdl.handle.net/11858/00-097C-0000-0022-FF60-B (visited on 12/16/2015).

Ulli Waltinger (2010). "GermanPolarityClues: A Lexical Resource for German Sentiment Analysis." In: *Proceedings of the International Conference on Language Resources and Evaluation*.

Sida I. Wang and Christopher D. Manning (2012). "Baselines and Bigrams: Simple, Good Sentiment and Topic Classification." In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Mengqiu Wang and Christopher D. Manning (2013). "Effect of Non-linear Deep Architecture in Sequence Labeling." In: *Proceedings of the 6th International Joint Conference on Natural Language Processing*.

Paul J. Werbos (1982). "Applications of advances in nonlinear sensitivity analysis." In: *System Modeling and Optimization*. Ed. by R. F. Drenick and F. Kozin. Vol. 38. Lecture Notes in Control and Information Sciences.

Casey Whitelaw, Navendu Garg, and Shlomo Argamon (2005). "Using appraisal groups for sentiment analysis." In: *Proceedings of the ACM CIKM International Conference on Information and Knowledge Management*.

Janyce M. Wiebe and Rada Mihalcea (2006). "Word Sense and Subjectivity." In: *Proceedings of the 21st International Conference on Computa-*

tional Linguistics and 44th Annual Meeting of the Association for Computational Linguistics.

Theresa A. Wilson, Janyce M. Wiebe, and Paul Hoffmann (2005). "Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis." In: *Proceedings of the Conference on Human Language Technology Conference and Empirical Methods in Natural Language Processing*.

Theresa A. Wilson, Janyce M. Wiebe, and Paul Hoffmann (2009). "Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis." In: *Computational Linguistics* 35.3, pp. 399–433.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin (2015). "Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation." In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

David Yarowsky (1992). "Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora." In: *14th International Conference on Computational Linguistics*.

David Yarowsky (1995). "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods." In: *33rd Annual Meeting of the Association for Computational Linguistics*.

Wenpeng Yin and Hinrich Schütze (2015). "Multichannel Variable-Size Convolution for Sentence Classification." In: *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*.

Hong Yu and Vasileios Hatzivassiloglou (2003). "Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Mo Yu and Mark Dredze (2015). "Learning Composition Models for Phrase Embeddings." In: *TACL* 3, pp. 227–242.

Matthew D. Zeiler, Marc'Aurelio Ranzato, Rajat Monga, Mark Z. Mao, K. Yang, Quoc V. Le, Patrick Nguyen, Andrew W. Senior, Vincent Vanhoucke, Jeffrey Dean, and Geoffrey E. Hinton (2013). "On rectified linear units for speech processing." In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao (2014). "Relation Classification via Convolutional Deep Neural Network." In: *Proceedings of the 25th International Conference on Computational Linguistics*.

Torsten Zesch and Iryna Gurevych (2006). "Automatically Creating Datasets for Measures of Semantic Relatedness." In: *Proceedings of the Workshop on Linguistic Distances*.

Zhihua Zhang, Guoshun Wu, and Man Lan (2015). "ECNU: Multi-level Sentiment Analysis on Twitter Using Traditional Linguistic Features and Word Embedding Features." In: *Proceedings of the 9th International Workshop on Semantic Evaluation*.

Geoffrey Zweig and Christopher J. C. Burges (2011). *The Microsoft Research Sentence Completion Challenge*.