Fachhochschule Dortmund

Fachbereich Informatik

Thesis zur Erlangung des akademischen Grades
Bachelor of Science
B. Sc.
im Studiengang Softwaretechnik-Dual

Sentiment Analysis Based on Word Embeddings: Possible Improvements and Transfer to the German Language

Leonard Hövelmann

geboren am 2. Oktober 1992

Matrikelnummer: 7091008

Betreuung durch

Prof. Dr. Christoph M. Friedrich

Zweitprüfer:

M. Sc. Sven Koitka

Dortmund, den 29.8.2017

Abstract

Multilingual text classification is a topic that incorporates the disciplines statistical learning, natural language processing, as well as statistics. In a first part of this thesis, the applicability of automatic text translation algorithms with respect to multilingual sentiment classification of natural language text is determined. Therefor a German text corpus consisting of product reviews, is presented and compared with a similar English corpus. Each corpus is translated into the other language using two different automatic translation algorithms. Subsequently, two different text classification algorithms are applied to the corpora, building different combinations of training and validation sets. In a second part, the participation in two subtasks of the *Germeval 2017 Task on Aspect Based Sentiment Analysis* is described: *document relevance* (task A) and*document-level sentiment classification* (task B). The best system proposed in this thesis achieved the second place of twelve systems submitted by seven teams for task A for both of the two test sets. For task B, the proposed system achieved the first place for test set 1 and the second place for test set 2 of 17 systems submitted by eight different teams. The two parts *text translation* and *GermEval-2017 participation* are discussed in two separate documents, respectively written in journal style, that are concatenated in this document.

Zusammenfassung

Mehrsprachige Textklassifikation ist ein Thema, das die Disziplinen des statistischen Lernens, der natürlichsprachigen Textverarbeitung sowie der Statistik vereinigt. Diese Thesis untersucht die Anwendbarkeit von automatischen Textklassifikationsalgorithmen hinsichtlich der mehrsprachigen Stimmungsklassifikation von natürlichsprachigen Texten. Dazu wird ein deutschsprachiger Textkorpus, bestehend aus Produktbewertungen, vorgestellt und einem ähnlichen, englischsprachigen Korpus gegenüber gestellt. Beide Korpora werden jeweils mit zwei verschiedenen automatischen Übersetzungsalgorithmen in die jeweils andere Sprache übersetzt. Basierend auf diesen Korpora werden dann zwei verschiedene Klassifikationsalgorithmen auf die Korpora angewandt, wobei jeweils unterschiedliche Kombinationen aus Trainings- und Validierungsset gebildet werden. Zusätzlich wird die Teilnahme am *Germeval 2017 Task on Aspect Based Sentiment Analysis* beschrieben, bei dem an den beiden Subtasks *Relevanz der Dokumente* (Task A) und *Sentiment Klassifikation auf Dokumenten-Ebene* (Task B) teilgenommen wurde. Für Task A hat das beste System jeweils Platz 2 von zwölf Systemen von sieben unterschiedlichen Teams für beide Test-Sets erreicht. Von 17 unterschiedlichen Systemen, die von acht Teams entwickelt wurden hat das beste in dieser Thesis vorgestellte System für Task B den ersten Platz für das erste Test-Set und den zweiten Platz für das zweite Test-Set erreicht. Die zwei unterschiedlichen Themengebiete *Textübersetzung* und *GermEval-2017 Teilnahme* werden in zwei unterschiedlichen Dokumenten, jeweils im Journal-Stil geschrieben, erläutert, die in diesem Dokument zusammengefügt werden.

Erläuterung zur Struktur

Diese Bachelorthesis besteht aus zwei unabhängigen Dokumenten, die jeweils zwei unterschiedlichen Gestaltungsrichtlinien folgen. Beide Dokumente wurden in diesem Dokument zusammengefügt, welches dem formalen, äußeren Erscheinungsbild einer Bachelorthesis genügt.

Das erste Dokument, *Sentiment Analysis Based on Word Embeddings: Possible Improvements and Transfer to German Language*, bildet den größeren Teil der schriftlichen Arbeit. Es ist für die spätere Veröffentlichung in einem IEEE-Journal vorgesehen und folgt deshalb grob der entsprechenden Gestaltungsvorschrift. Der Abschnitt *Share of Common Words* des Dokuments wird nicht Bestandteil der eigentlichen Publikation sein. Er ist deshalb in einem Appendix an das Dokument angehängt.

Das zweite Dokument, *Fasttext and Gradient Boosted Trees at GermEval-2017 Tasks A and B on Relevance Classification and Document-level Polarity*, beschreibt die Teilnahme am GermEval-2017 Workshop. Es folgt deshalb den entsprechenden Gestaltungsrichtlinien. Dieses Dokument hat ebenfalls einen Appendix, der nicht Teil der eigentlichen Einreichung war.

Sentiment Analysis Based on Word Embeddings: Possible Improvements and Transfer to German Language

Leonard Hövelmann

1 Introduction

Recently, automatic text classification and sentiment analysis gained attention due to their potential for industrial applications (e.g. customer relation management). During the past years, lots of different approaches were examined ranging over a large variety of machine learning algorithms and natural language processing (NLP) applications. However, most of these approaches were developed for English language. The behavior of many algorithms applied on German texts is not well researched. Moreover, text classification algorithms cannot be applied to German texts whenever an algorithm makes use of English word lists.

A lot of labeled training data is derived from sources in English language. It is desirable to make use of the large amount of annotated training data, even if the to-be-classified targets use a different language. In those cases, automatic machine translation could be helpful to solve the disparity in language between training and target data. Many recent works have proposed systems that are able to classifiy text in different languages with increasing classification performance. They are discussed in section 2.4. Some of these algorithms, however, have some disadvantages. Deep learning approaches require a large training set since they have a very large number of free parameters. Other approaches are limited to the languages they have been trained on. If text that originates from another language shall be classified, a new system has to be trained. Moreover, only few algorithms are implemented with a sufficient performance to be used for industrial purposes. Hence, there are cases in which a monolingual text classification algorithm is more appropriate. However, often the performance of a classifier decreases if a translated corpus is used for training or testing instead of a native lingual corpus.

This paper discusses techniques to shrink this performance gap. For this purpose, the fastText classifier and a multilayer perceptron are exposed to a bilingual text classification scenario by using two different machine translation methods. The fastText classifier is based on word embeddings while the multilayer perceptron is trained on bag-of-words (BOW) word vectors. Furthermore, a new German text corpus for sentiment classification is presented and translated to English language using two different methods. Similarly, an existing English corpus is translated to German using two different methods. Based on these corpora, the classifiers are evaluated with respect to their intra-lingual classification performance and their ability to improve the inter-lingual classification performance by augmenting the training dataset with translated texts.

Section 2 gives an introduction to the subject areas word embeddings, domain adaptation, sentiment analysis, and cross-lingual text classification and the respective state-of-the-art. Section 3 gives an description of the used corpora and of the experimental setup. Section 4 summarizes the results of the experiments. The discussion on the results is given in section 5 and section 6 shows a conclusion of the results.

2 Previous Work

The following section gives a short introduction to the techniques that are used in the experiments. Moreover, the state-of-the-art is given for each subject area.

2.1 Word Embeddings

Word embeddings are a technique to create vector representations for words. In contrast to other techniques like BOW, they can yield relatively low dimensional word vectors which is favorable in many cases. Typical dimensions for embedded word vectors are 100 to 300. Word embeddings are based on the idea that the probability of a word can be predicted given its predecessors in a document [1]:

$$p(w_{T+1}) = \prod_{t=1}^{T} p(w_t | w_1^{t-1}) \tag{1}$$

where w_t is the t-th word and $w_i^j = (w_i, w_{i+1}, \ldots, w_{j-1}, w_j)$. However, incorporating all the previous words for computing the probability of the next one (see eq. 1) presents a drawback in terms of computational efficiency. Therefore, most models suppose that only the last n words are sufficient to accurately compute the probability of the successive word [1] :

$$p(w_t | w_1^{t-1}) \approx p(w_t | w_{t-n+1}^{t-1}) \tag{2}$$

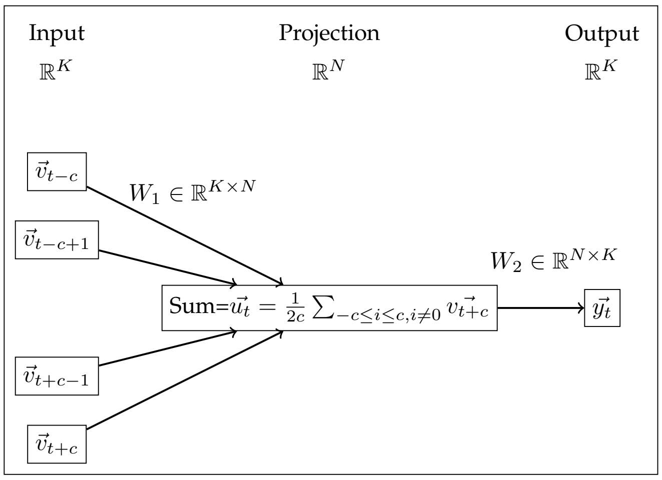
Input
$\mathbb{R}^K$
Projection
$\mathbb{R}^N$
Output
$\mathbb{R}^K$
$\vec{v}_{t-c}$
$\vec{v}_{t-c+1}$
$\vec{v}_{t+c-1}$
$\vec{v}_{t+c}$
$W_1 \in \mathbb{R}^{K\times N}$
$W_2 \in \mathbb{R}^{N\times K}$
Sum=$\vec{u_t} = \frac{1}{2c}\sum_{-c\leq i\leq c, i\neq 0} \vec{v_{t+c}}$
$\vec{y_t}$

Fig. 1. The CBOW Model

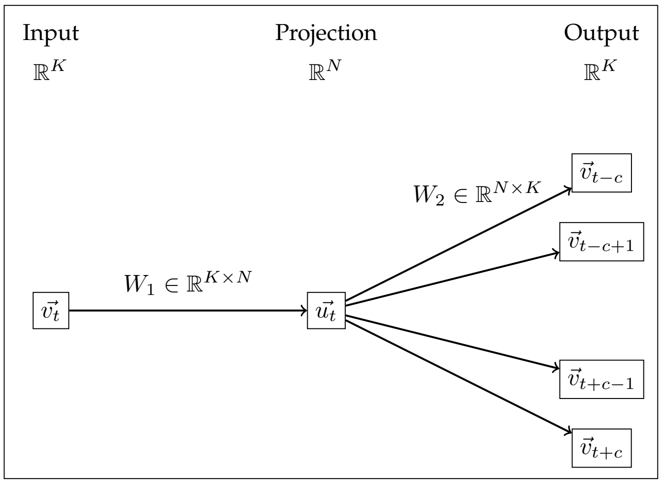

Fig. 2. The Skip-gram Model

That idea was generalized in later applications, for example by predicting the probability of a word given the previous but additionally also the following words or by predicting the context of a word given a word. The words surrounding the current word of interest are often called *context window*. Mikolov et al. [2] proposed two different ways of determining word vector representations: continous bag of words (CBOW) and the Skip-gram model which will be explained in the following paragraphs.

The CBOW model (figure 1) predicts a word given its context. Given a set of words $w_0, w_1, \dots, w_{|V|}$ in a vocabulary V, the training objective is to maximize the log probability

$$\frac{1}{|V|}\sum_{t=1}^{|V|}\sum_{-c\leq i\leq c, i\neq 0} \log p(w_t|w_{t-c},\dots,w_{t-1},w_{t+1},\dots,w_{t+c}) \tag{3}$$

The Skip-gram model, however, is trained to predict a context, given the word:

$$\frac{1}{|V|}\sum_{t=1}^{|V|}\sum_{-c\leq i\leq c, i\neq 0} \log p(w_{t+i}|w_t) \tag{4}$$

Hence, the Skip-gram model yields $2c$ different distributions whereas the CBOW model yields only one distribution.

The working principle of word2vec is similar to an autoencoder. First, input is projected to a lower dimensional hidden layer. Then, the model is trained to minimize the discrepancy between the reconstruction of the condensed input and the training objective that is passed to the loss function. In contrast to an autoencoder, however, word2vec does not try to reconstruct the input itself. If the input is the context of a word, it tries to reconstruct the target word (CBOW), whereas it tries to reconstruct its context (Skip-gram) if the input is a single word. In the CBOW setting, the context words are averaged first and subsequently projected onto the hidden layer. Due to this procedure, the word order information is lost, hence the notion *bag-of-words* in its name.

The different word vectors $\vec{v_j}$ are one-hot-encoded vector representations. Therefore, a word w_j (Skip-gram) or the context of a word (CBOW) can be transferred to its hidden layer representation by using a weight matrix $W_1 \in \mathbb{R}^{K\times N}$ where N is the dimensionality of the hidden layer and $K = |V|$. Also, the hidden layer size is the size of the word vectors the model produces. Then, the hidden layer representation is projected into the output layer using a second weight matrix $W_2 \in \mathbb{R}^{N\times K}$.

The Skip-gram model is trained by maximizing equation 4 where $p(w_{t+i}|w_t)$ can be computed using the softmax function:

$$p(w_{t+i}|w_t) = p(w_O|w_I) = \frac{e^{{v'}_{w_O}^{\intercal} v_{w_I}}}{\sum_{w=1}^{|V|} e^{{v'}_{w}^{\intercal} v_{w_I}}} \tag{5}$$

where v'_{w_i} is the *output representation* of word w_i [3]. The *output representation* is the vector that is produced by applying the two matrix multiplications to a one-hot-encoded input representation. In practice, this way of computing $p(w_{t+i}|w_t)$ is disadvantageous because it requires to iterate over the whole vocabulary. A hierarchical version of the softmax, introduced by Morin and Bengio [4], requires only $\log_2 |V|$ iterations. The softmax function guarantees the probability of all words amounting to 1. This property is often required in cases when the softmax function is used as an activation function for the last layer of a neural network. In such cases the normalization is inevitable and to get to the relative probability of a certain class is required. The Skip-gram model, however, "is only concerned with learning high-quality representations" [2]. Hence, Mikolov et al. [2] introduced *negative sampling*, which still ensures high-quality vector representations by using a simplified training objective.

The *negative sampling* training objective for a word pair (w_O, w_I) is defined as follows [2]:

$$\log \sigma(v_{w_O}^{'\intercal} v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)}[\log \sigma(-v_{w_i}^{'\intercal} v_{w_I})] \tag{6}$$

where $P_n(w)$ is the *noise distribution* that generates wrong samples and σ is the logistic function

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{7}$$

This training objective maximizes the probability of a pair coming from the training data by maximizing the probability that wrong samples do *not* come from the training data.

Word2vec has shown to capture syntactic regularities that can be represented by vector arithmetic. Global vectors for word embeddings (GloVe) [5] is another word embedding technique that aims to explicitly capture these lingual regularities of word2vec. The optimization of GloVe is driven by the discovery, that the ratio of co-occurrence probabilities is a better measure for word similarities than the co-occurrences itself.

2.1.1 fastText

fastText [6] is the name of a text classification system but also that of a word embedding technique. Its setup is inspired by the CBOW model [2] and driven by the goal to develop a fast text classification method since models based on neural networks "tend to be relatively slow both in train and test time" [6]. In contrast to the Skip-gram or the CBOW architecture [3], fastText also takes character-level n-grams into account. In the fastText model, words are represented by the sum of their character level n-grams. For example, the word *human* is subdivided into the character-level trigrams

<hu, hum, uma, man, an>

and the term "<human>" itself, where $<$ and $>$ are special characters used by fastText to indicate the beginning or the ending of a word. The character-level n-grams themselves are represented by vectors. Hence, the representation of a whole word can be obtained by the sum of the vector representations of the character-level n-grams.

When using fastText for classification instead of prediction a word the goal is to predict a class (or label) - for example *positive* or *negative* in the sense of sentiment polarity. Thus, the output $\vec{y_t}$ would be replaced by a class 0 for negative sentiment and 1 for positive sentiment. Similar to the CBOW model, fastText works with language-dependent word embeddings. To represent a document, word embeddings are averaged which results in a loss of word order. The model is capable of handling sentences with a varying number of words. For classification, fastText also uses bigram features in order to retain information about the word order. Keeping that information is able to improve classification performance in sentiment analysis tasks [7]. The model is trained by minimizing the negative log-likelihood over classes

$$-\frac{1}{N} \cdot \sum_{n=1}^{N} y_n \cdot \log(f(B \cdot A \cdot x_n)) \tag{8}$$

where x_n "is the normalized bag of features of the n-th document" [6], y_n the label, and A and B are weight matrices. fastText can be trained with different loss functions: softmax, hierarchical softmax, and negative sampling (default setting[1]).

2.2 Domain Adaptation

In general, the dataset for training a classifier is derived from a particular domain equal to the domain of its target application, for example book reviews. Let $\mathcal{X}$ be the set of training features and $\mathcal{Y}$ the set of class labels where $\{x_i, y_i\}_{i=0}^{N}, \quad x_i \in \mathcal{X}, \quad y_i \in \mathcal{Y}$ is an element in the labeled training data. If all elements (x_i, y_i) in the training and the test dataset are randomly derived from an unknown probability distribution p, there is no reason to apply domain adaptation. The necessity to apply it only arises under the following conditions. The observed probability distribution p_s differs significantly from p. Hence, a classifier trained on training data derived from p_s can only reconstruct p_s. When such a classifier is applied to a new domain with probability distribution $p_t, \quad p_t \neq p_s$, of which only the features are known, the classifier will show bad performance since it tries to reconstruct p_s for samples that are derived from p_t.

Blitzer et al. [8] developed structural correspondence learning (SCL) to perform domain adaptation for part-of-speech tagging by predicting frequently occurring *pivot features* in the target domain. This algorithm was refined for sentiment analysis problems (SCL-MI) [9]. In essence, the SCL algorithm shifts p_t to p_s.
By augmenting the feature space using linear kernels, Daumé [10] built a classifier that is able to handle domain adaptation. Just like SCL, it requires labeled features in the target domain.
Glorot et al. [11] used Stacked Denoising Autoencoders (SDA_{SH}) to build a classifier that is robust towards noise on data. They used an autoencoder that is trained to reconstruct the original signal, given a noisy version of itself [12]. In a second step, they train a support vector machine (SVM) on the transformed data. In contrast to SCL, this algorithm performs no adaptation for a specific domain, but tries to generalize to arbitrary domains. By perturbing the data, p_s is less biased regarding p.
Bollegala et al. approached domain adaptation for sentiment analysis using a *sentiment sensitive thesaurus* [13]. They used labeled training data to identify elements with a high point-wise mutual information with the class labels and created a thesaurus that holds informations about the occurrence of such elements in a review. In a second step, the augmented data from an unknown domain with elements from this thesaurus.
Bi-Transferring Deep Neural Networks (BTDNNs) [14] are also based on the idea of autoencoders. In contrast to SDA_{SH}, BTDNNs perform an explicit domain adaptation. They use one encoder to map input to a latent feature representation and two decoders to decode this representation in terms of the source or the target domain. The encoder

1. https://github.com/facebookresearch/fastText, last access: 08/24/2017

and decoders are trained defining a training objective that allows reconstructing the source and the target domain from each other besides the common training objective of autoencoders.

2.3 Sentiment Analysis

Sentiment analysis algorithms can operate on three levels: document level, sentence level, and aspect level [15, p. 9].

Aspect level sentiment analysis identifies the sentiment regarding different aspects occurring in a text. The sentence *"I love riding my beautiful new car across the bumpy road."* for example expresses two different sentiments regarding two aspects. In terms of the aspect *car*, the author expresses a positive sentiment. Referring to *road*, however, he expresses negative sentiment. Aspect level sentiment analysis requires the identification of the opinion target as well as the opinion itself. It is particularly useful in cases where various opinion targets of interest are present within a sentence.

Sentence level sentiment analysis requires a less fine-grained approach. In general, sentiment analysis on sentence levels distinguishes three different sentiment classes: *positive*, *neutral*, and *negative* since not every sentence expresses any sentiment [15, p. 9]. This step expects the identification of particular sentences.

Document level sentiment analysis incorporates whole documents. A popular example for this level is product reviews. In product reviews, the whole document can be considered as an expression of sentiment with respect to a single aspect - the product. Multiple entities in a document exceed the capability of the document level sentiment analysis . Often, it is only distinguished between positive and negative opinions, based on the assumption that reviewers generally do not express a neutral sentiment about a product.

In the remainder of this paper, the term *sentiment analysis* refers to document level sentiment analysis since all following applications work with product reviews.

The usage of recurrent neural networks for natural language processing was examined by Socher et al. [16].

The field of sentiment analysis is strongly connected to text classification since sentiment analysis is often understood as a specialized text classification task. An early approach to perform sentiment classification with statistical learning methods was presented by Pang et al. [17]. Using convolutional neural networks (CNNs), Yoon Kim [18] built a sentence level sentiment classifier. Kalchbrenner et al. proposed a similar approach [19]. Text classification using CNNs that operate on character-level were proposed by Zhang et al. [20]. Over the past years, Sentiment analysis on microblogs such as Twitter gained lots of attention [7], [21], [22], [23], [24]. The influence of different resources on the classification performance of Tweets was examined by Dalmia et al. [25]. Recurrent neural networks (RNNs) have shown to be successful for text classification [26], [27], [28]. Yang et al. [29] proposed a text classification system that operates with hierarchical recurrent neural networks and the attention mechanism.

2.4 Cross-Lingual Text Classification

The design of cross-lingual text classifiers aims at good classification performance in multiple languages. Simple BOW vectors are inapt for this purpose. Other techniques should supplant this approach that capture more efficiently the relations between words. The alignment of words produced by the CBOW or Skip-gram models [2], is quite similar for different languages.

Based on this discovery, Mikolov et al. proposed a system to find a linear projection between vector spaces of different languages using a bilingual dictionary [30]. They assume that it is possible to transform the vector space of a model trained in one language into the vector space of another language with a linear projection. The linear projection is performed by a translation matrix, whose weights are determined by utilizing the bilingual dictionary:

$$\min_{w} \sum_{i=1}^{n} ||Wx_i - z_i||^2 \tag{9}$$

where x_i is the word embedding representation of word i in the source language and z_i is the word embedding representation of word i in the target language.

The objective function for learning the word embeddings is based on maximum likelihood, while the distance function is the cosine distance and the loss function to determine the linear projection is the mean squared error [30]. This irregularity led Xing et al. [**?**] to the development of normalized word embeddings and a different computation of the projection matrix.

Instead of transforming the vector space representation of the source language into the vector space representation of the target language, Lauly et al. [31] pursued another approach. They trained autoencoders to directly transform the BOW representation of one sentence in the source language into the BOW representation in the target language. To increase computational efficiency, they used a hierarchical word representation. For encoding, the regular BOW representation is chosen. No training is performed during this step - the term frequency - inverse document frequency (TF-IDF) [32] BOW representation of a word builds the hidden layer. Since the vocabulary size can differ between two languages, they are adjusted such that the hidden layers always have the same size. For decoding, two decoders are trained - one to decode into language $\mathcal{X}$ and the other to decode into language $\mathcal{Y}$. For training the decoders, the Europarl corpus [33] was used.

Using SCL, Prettenhofer et al. developed cross-language classification using structural correspondence learning (CL-SCL), an extension that performs multilingual sentiment classification [34]. For this purpose, they search for words that contain high class information with positive sentiment and frequent occurrance. Having these potential high-quality class predictors, they identify pivot features by

TABLE 1
Number of Reviews in the Different Domains (German Corpus)

Domain	# Reviews	Polarities		
		# pos	# neutral	# neg
G	2221	1146	114	961
D&B	1663	887	81	695
C&P	357	239	26	91
PN	732	567	56	108
K&H	1712	962	111	639
SP	641	447	27	167
T&HI	1966	1299	114	553
B	3384	2599	274	499

building word pairs of a word from the source domain and its translation in the target domain.

The viability of using machine translation to transfer a corpus from one language into another was examined by Banea et al. [35] by comparing different classifiers on different language resources. Using English sentiment words and their Chinese translations, Zhou et al. [36] created bilingual sentiment word embeddings by using denoising autoencoders. A similar approach was pursued by Zhou et al. [14], who created bilingual word embeddings, though not exclusively for sentiment words. Also by using a translated corpus, Zhou et al. [37] jointly trained two bidirectional LSTMs using the attention mechanism. Using character-level embeddings and convolutions, Wehrmann et al. [38] built a multi-lingual text classifier with low memory consumption that requires no aligned or translated data.

3 Experimental Setup

With the goal to examine the influence of using translated language for a text classifier instead of native language, different experiments were set up.

First, the influence of automatic text translations regarding the quality of text classifiers is examined. Therefor, a German and an English text corpus containing labeled product reviews, were translated. Each corpus was translated to the other language using an open source translation program and a commercial translation API, whereas due to technical reasons two different commercial APIs were used.

Second, the possibility of enhancing a classifier by using translated texts for training in addition to native lingual texts was investigated. For this purpose, classifiers were trained on a merged dataset built from native lingual and translated reviews. The performance of these classifiers is compared to classifiers that were trained exclusively on native lingual reviews.

3.1 Corpora

At least four different corpora are required to perform the previously-described experiments: (1) a German corpus, (2) the translation of a German corpus, (3) an English corpus, and (4) the translation of an English corpus. In order to compare different translation algorithms, six corpora were created in total with the two additional corpora originating from a different translation algorithm. The translations are described in subsection 3.4.

TABLE 2
Number of Reviews in the Different Domains (English Corpus)

Domain	# Reviews	Polarities		
		# pos	# neutral	# neg
P,L&G	12550	1119	1555	9876
M&T	87559	7949	7802	71808
PN	95208	10951	9345	74912
K&H	96278	9402	6945	79931
T&HI	95795	7278	7418	81099
B	94097	9036	9782	71679

TABLE 3
Domain Statistics for Review Lengths (German Corpus, in Characters)

Domain	mean	median	σ	min	max
G	448.4	859	584.9	21	14448
D&B	674.5	223	1009.5	86	11983
C&P	2049.6	1801	2475.9	73	22097
PN	547.5	595	622.0	68	7937
K&H	634.6	374	813.8	42	9091
SP	326.9	136	274.6	18	2299
T&HI	564.0	342	746.0	50	10084
B	597.5	209	775.6	15	9604

TABLE 4
Domain Statistics for Review Lengths (English Corpus, in Characters)

Domain	mean	median	σ	min	max
P,L&G	829.8	590	807.5	1	12995
M&T	893.4	489	1110.1	1	23095
PN	441.0	288	489.9	1	15621
K&H	499.9	317	569.5	1	30802
T&HI	536.0	324	645.6	1	19242
B	834.5	469	1011.1	1	30444

TABLE 5
Domain Statistics for Review Lengths (German Corpus, in Words)

Domain	mean	median	σ	min	max
G	68.9	128	109	4	3374
D&B	104.9	36	154.7	19	1998
C&P	313.1	293	371.2	22	3283
PN	84.1	93	94	21	1213
K&H	97.4	57	122.4	9	1411
SP	51.8	22	42.8	5	382
T&HI	85.9	57	112.5	17	1468
B	91.6	32	118.6	4	1510

TABLE 6
Domain Statistics for Review Lengths (English Corpus, in Words)

Domain	mean	median	σ	min	max
P,L&G	156.9	113	150.0	1	2345
M&T	159.3	90	193.24	1	3874
PN	85.8	57	92.7	1	3034
K&H	95.1	61	105.7	1	4995
T&HI	101.3	62	119.4	1	3548
B	148.8	86	174.8	1	4978

TABLE 7
Domain Statistics for Review Lengths (German Corpus, in Sentences)

Domain	mean	median	σ	min	max
G	4.9	4	5.3	1	122
D&B	6.3	4	8.3	1	116
C&P	18.5	11	21.4	1	198
PN	5.2	4	5.3	1	60
K&H	6.6	4	7.5	1	76
SP	4.0	3	3.0	1	26
T&HI	6.0	4	7.1	1	87
B	5.7	4	6.8	1	84

TABLE 8
Domain Statistics for Review Lengths (English Corpus, in Sentences)

Domain	mean	median	σ	min	max
P,L&G	7.4	6	6.4	1	107
M&T	7.2	5	7.7	1	163
PN	4.7	4	4.6	1	721
K&H	5.2	4	4.6	1	209
T&HI	5.4	4	5.2	1	122
B	6.8	5	6.8	1	194

TABLE 9
Vocabulary Statistics (German Corpus)

Domain	Unigrams	Bigrams	Trigrams
G	16563	87428	135,581
D&B	20584	101,900	157,372
C&P	13166	68673	106,187
PN	9056	40544	57530
K&H	16942	94938	150,297
SP	5883	25203	35248
T&HI	17842	97962	152,795
B	28495	161,253	268,340

TABLE 10
Vocabulary Statistics (English Corpus)

Domain	Unigrams	Bigrams	Trigrams
P,L&G	60,013	501,971	1,230,839
M&T	34,0961	2,989,613	7,777,735
PN	135,427	1,270,427	3,830,130
K&H	162,848	1,456,418	4,329,004
T&HI	188,775	1,657,037	4,821,108
B	299,497	2,750,034	7,323,181

The German corpus consists of amazon product reviews that have been selected in a time period lasting from 11/18/2016 to 02/20/2017. The reviews were taken from the following eight product categories: “Garden” (G), “Camera and Photo” (C&P), “Pet Nutrition” (PN), “Tools and Home Improvement” (T&HI), “Sports Products” (SP), “Kitchen and Household (K&H)”, “Books” (B), and ”DVDs and Blu-Rays” (D&B). The product categories were customarily selected on the basis of the category overview of the German amazon website[2] on that time. Due to legal issues, the original reviews cannot be published. However, a list of the product and review IDs as well as a tool to download the original reviews is provided[3].

2. https://www.amazon.de/gp/site-directory/ref=nav_shopall_btn, last access: 06/21/2017

TABLE 11
Mapping of Domain Labels Between English and German Corpus

Domain Label in English Corpus	Domain Label in German Corpus
Books	Books
Kitchen	Kitchen & Household
Movies & TV	DVDs & Blu-Rays
Patio, Lawn & Garden	Garden
Pet Supplies	Pet Nutrition
Tools & Home Improvement	Tools & Home Improvement

In addition to the German review corpus, English amazon product reviews have been used from an existing corpus[4] [39]. From this corpus, reviews from domains similar to the ones in the native lingual corpus were selected (see Table 11). Based on these domains, random samples were translated. The mapping between the domains is shown in Table 11.

Tables 3 to 10 give detailed statistics of the two corpora. Reviews with five or four stars are considered as positive (“pos”), reviews with three stars as neutral, and reviews with one or two stars are considered as negative (“neg”). Neutral reviews were omitted, only positive and negative reviews were used for the following experiments. Table 3 provides information about the review length in characters. Table 9 shows the number of disjoint words in each domain.

3.2 Text Preprocessing

Among two commercial translation APIs, the Moses translator was used to translate the reviews (see subsection 3.4). The Moses translator itself is not able to distinguish different sentences, therefore the texts have to be split into single sentences and fed into the translator individually. Hence, the reviews had to be preprocessed before being translated with Moses. First, the text was transformed to lower case. Since the translation is used for classification only, and lower-casing is a preprocessing step for any used classifier, the casing tool of Moses remains unused. The punctuation marks “.”, “!”, and “?” were replaced with a line break. Other punctuation marks (;,()[]$-_—%§*+-# /) were removed. English abbrevations like “I’m” should be forwarded in their long form to the translator, therefore the following replacements were performed: “’s” was replaced with ”is”, “i’m” was replaced with ”i am”, ”n’t” was replaced with ”not”, “’ll” was replaced with ”will”, and “’ve” was replaced with “have”.
For classification, the text is preprocessed by transforming to lower-case and stemming the text using the German variant of the Snowball stemmer[5].

3. https://github.com/LeoIV/ReviewCrawler, last access: 08/29/2017, license: MIT
4. http://jmcauley.ucsd.edu/data/amazon/, last access: 06/29/2017
5. http://snowballstem.org/, last access: 06/21/2017, license: 3-clause BSD

3.3 Classifier

Two different classifiers were used: a feedforward multi-layer perceptron (MLP) and fastText. The feedforward MLP has one hidden layer. The output layer is trained using the softmax function while the hidden layer uses the sigmoid function. The number of output units varies depending on the number of classes. The network can be described as

$$\vec{y} = \sigma(\vec{w}_2^T \cdot \text{sig}(\vec{w}_1^T \cdot \vec{x} + b_1) + b_2) \tag{10}$$

with

$$\sigma(\vec{x})_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}, \quad i = 1, \ldots, K \tag{11}$$

$$\text{sig}_j(t) = \frac{1}{1 + e^{-t_j}} \tag{12}$$

where K is the number of classes and $\text{sig}_j(t)$ is the value for the j-th hidden unit.

Additionally, fastText [6] is used as a word-embedding-based algorithm.

3.4 Translation

Three different algorithms were used for translation. Moses [40] is an open source toolkit for statistical machine translation[6] (SMT), which is based on phrase-based statistical machine translation.

In phrase-based SMT, instead of aligning words, an algorithm searches for matches of phrases in the source and the target language. $(\bar{f}, \bar{e})$ is a phrase pair, where $\bar{e}$ consists of the words $e_0, \ldots, e_i$ and $\bar{f}$ consists of the words $f_0, \ldots, f_j$. The phrase-based SMT model consists of three stages: a phrase translation table, a reordering model, and a language model [41, p. 136] where the phrase translation table holds scores

$$\Phi(\bar{f}|\bar{e}) = \frac{\text{count}(\bar{e}, \bar{f})}{\sum_{\bar{f}_i} \text{count}(\bar{e}, \bar{f}_i)} \tag{13}$$

for a certain combination of phrases, the reordering model $d(x) = \alpha^{|x|}, \quad \alpha \in [0, 1]$ computes the costs for a reordering of phrases, and the language model $p_{\text{LM}}(e)$ ensures “fluent” language. The costs of the reordering model d are based on the *movement* that have to move a phrase in a particular sentence [41, p. 130]. The language model is defined as shown in equation 1 with $w_0, \ldots, w_{T+1} = e_0, \ldots, e_i$. Using these definitions, the phrase-based SMT model can be defined [41, p. 136] as

$$e_{\text{best}} = \arg\max_e \sum_{i=1}^{I} [\Phi(\bar{f}_i|\bar{e}_i) d(\text{start}_i - \text{end}_{i-1} - 1)] p_{\text{LM}}(\bar{e}) \tag{14}$$

where e_{best} is the best translation according to the model, I is the number of phrases in the model, and start_i is “the position of the first word of the foreign input phrase that translates to the i-th English phrase, and end_i is the position of the last word of that foreign phrase” [41, p. 129].

6. http://www.statmt.org/moses/, last access: 07/21/2017, license: LGPL

Moses comes with various pre-trained models, trained on the Europarl corpus[7] [33]. Both the German-English as well as the English-German model were used for translation. Typical translations generated by Moses are:

> *“bitterly at the beginning , i thought that there was a fine levers , and i have also very reingesteigert therefore , it is rather i am told marks a musiklehrer because , at some point , the point we here in this matter stuck we can implement it properly in this way , there are thousands of hefte over the nix”*

in case of German to English translation, or

> *“dies ist ein liebenswürdiger version der klassischen dicken ist märchen henry winkler ist eine gute aufzuzeigen , wie die "scrooge" charakter obwohl sie wissen , was passieren wird , diese fassung hat genug von einer änderung , um sie besser zu machen , dass die durchschnittliche wenn sie die liebe ein weihnachtslied in jedem version , dann werden sie lieben das”*

in case of English to German translation. The text is hardly understandable to a human reader. Many words cannot be aligned and therefore remain in their original language.

For English to German translation, the Google Cloud Translation API[8] was applied to a subset of the English dataset. According to the Google Blog[9], the German to English translations are based on neural machine translation (NMT) [42], [43]. Googles NMT algorithm uses recurrent neural networks (RNNs) to perform the translation. The basic idea is based on the Encoder-Decoder architecture, which is an extension of RNNs that enables them to handle input sequences whose length differs from the length of the output sequence [44], [45]. The Encoder-Decoder architecture is enhanced by the *attention mechanism* [46] that learns to associate the output of the encoder to the elements of the output of the decoder. Wu et al. used eight stacked long short-term memory (LSTM) [47] layers for encoding as well as for decoding, whereas the first layer of the decoder is a bidirectional LSTM layer.

For German to English translation, the Yandex Translation API was used[10]. Just like Moses, the Yandex Translation API is phrase-based[11].
A typical translation generated by the Yandex Translation API is:

7. http://www.statmt.org/moses/RELEASE-3.0/models/, last access: 07/31/2017
8. https://cloud.google.com/translate/, last access: 07/31/2017, license: commercial with a free tier
9. https://www.blog.google/products/translate/found-translation-more-accurate-fluent-sentences-google-translate, last access: 08/23/2017
10. https://tech.yandex.com/translate/, last access: 07/31/2017, license: commercial with a free tier
11. https://yandex.com/company/technologies/translation/, last access: 08/23/2017

"Super Book!!!! I finally strapped the notes. The book is clearly written and very logically, so that you will be really guided step-by-step approach the subject. Absolute recommendation for all beginners, the notes want to learn to read!"

and a typcial Google Cloud API translation is:

"Perfekt für kleine Hunde und Käfige! Wir haben einen kleinen Chihuahua unter 5 lbs. Zwei dieser Schalen passen perfekt in ihrem Käfig mit ihr, und ich habe keinerlei Beschwerden mit diesen Schalen hatten. Sie reinigen sich gut und haben keine Anzeichen von Rost oder erhebliche Mängel gezeigt. Ich würde diese sofort wieder mit Null zu zögern kaufen. Absolut perfekt!!!"

These reviews are easier to read and no words remain untranslated.

In the following, the different translation algorithms are compared in terms of their suitability for sentiment classification. In case of English to German translation, Moses is compared to the Google Cloud Translation API while in case of German to English translation, Moses is compared to the Yandex translation API.

3.5 Influence of Translations

In this setup, the influence of using translations instead of native lingual reviews is examined. For this reason, the following system runs were executed.

First, the two different classifiers as explained above are trained and tested on native lingual reviews in both German and English language.
Second, the classifiers are trained on the native lingual reviews, and tested on the translated reviews. This is done for both directions: German to English and English to German as well as for the two different translations algorithms explained above using the two different classifiers.
Third, the two classifiers are trained and validated only on the translated algorithms.

To perform these experiments, six different datasets are required: (1) the native lingual German reviews, (2) the native lingual English reviews, (3) the German to English translated reviews using Moses, (4) the German-to-English translated reviews using the Yandex Translation API, (5) the English-to-German translated reviews using Moses, and (6) the English to German translated reviews using the Google Cloud Translation API.

Since these datasets originate from different sources, they vary with respect to the number of reviews they contain. To achieve comparability, they are size-adjusted. For all experiments explained in this subsection, the number of reviews in each dataset is shrunken to the smallest number of reviews in any dataset.

The experiments are validated using 10-fold cross-validation. In cases where the training dataset differs from the validation dataset, both are split into ten subsets. For each run, 90% of the training data is used for training, omitting 10% for the data where the 10%-split differs from run to run. 10% of the validation data is used for validation, using varying splits from run to run.

3.6 Augmenting Datasets

The following experiments are intended to clarify in which way a classifier, performing in one language, can be enhanced by augmenting a training set with additional reviews translated from another language. For example, it should be examined whether it is possible to enhance a binary text classifier, that performs in the English language, by augmenting its training set with reviews that were translated from German to English using Moses.

In this scenario, the following experiments are distinguished. The first two experiments follow this procedure: first, training a classifier on native lingual German reviews plus English to German translated reviews for both Moses and Google Cloud Translation API. Second, validating on native German reviews. Accordingly, the other two experiments train a classifier on native lingual English reviews plus German to English translated reviews for both, Moses and Yandex translation API and validate them on native lingual English reviews. The size of the datasets used for augmentation is set to the minimum of the respective translation. For example, when translated English reviews are used for augmentation and the length of the Moses translation corpus is less than the length of the Yandex translation corpus, a random sample with the length of the Moses corpus is taken from the Yandex corpus.

3.7 System Description

The experiments were run on Spark using Scala for the implementation. For running fastText, the Scala process package was used that can run external processes and handle their input and output.

The experiments have been executed on a Unix Machine with four Intel Xeon E5-2687W v3 CPUs. In total, the machine has 20 physical CPU cores and 256GB RAM. Of these 20 physical cores, at most, 12 were used since the machine is used by others as well.

4 Results

This section gives the results of the previously-described translation experiments. The scores are expressed in terms of the micro averaged F_1 scores of the different classifications or further computations on these scores.

4.1 Influence of Translations: Results

Tables 12 and 13 show the results of the experiments. For both German and English, and for all combinations (native to native, native to translated, translated to native, and

TABLE 12
Micro-Averaged F_1 Results of English as the Native Language

Training	Testing	fastText	MLP
native	native	0.896(±0.008)	0.879(±0.009)
		Moses	
		fastText	MLP
translated	native	0.848(±0.008)	0.804(±0.014)
native	translated	0.795(±0.010)	0.721(±0.017)
translated	translated	0.902(±0.007)	0.879(±0.010)
		Yandex Translation API	
		fastText	MLP
translated	native	0.853(±0.008)	0.809(±0.011)
native	translated	0.801(±0.018)	0.735(±0.013)
translated	translated	0.909(±0.007)	0.886(±0.008)

TABLE 13
Micro-Averaged F_1 Results of German as the Native Language

Training	Testing	fastText	MLP
native	native	0.917(±0.008)	0.891(±0.010)
		Moses	
		fastText	MLP
translated	native	0.794(±0.013)	0.688(±0.018)
native	translated	0.814(±0.009)	0.782(±0.021)
translated	translated	0.833(±0.011)	0.799(±0.015)
		Google Cloud Translation API	
		fastText	MLP
translated	native	0.881(±0.011)	0.799(±0.015)
native	translated	0.822(±0.008)	0.764(±0.014)
translated	translated	0.880(±0.009)	0.838(±0.011)

TABLE 14
Transfer Error for English: the transfer error is the discrepancy between the micro-averaged F_1 scores of the classifier trained on native lingual reviews and the classifier in the respective row. The significance is determined by a two-sided paired Wilcoxon-Mann-Whitney test for these two distributions with $n = 10$

		Moses		
Training	Testing	fastText	MLP	Sig.
translated	native	0.048(±0.005)	0.074(±0.017)	***
native	translated	0.101(±0.009)	0.158(±0.021)	***
		Yandex Translation API		
Training	Testing	fastText	MLP	Sig.
translated	native	0.043(±0.010)	0.070(±0.017)	***
native	translated	0.095(±0.018)	0.144(±0.017)	***

translated to translated), and for each classifier, (MLP and fastText), the results are listed. Table 12 shows the results of the experiments with English as the source language. Similarly, Table 13 shows the results with German as the source language. For example, the value *translated* in the column *Training* of Table 13 indicates that the model was trained on reviews translated from English to German. Accordingly, *native* in the column *Testing* means that the model was trained on native lingual reviews.

TABLE 15
Transfer Error for German: the transfer error is the discrepancy between the micro-averaged F_1 scores of the classifier trained on native lingual reviews and the classifier in the respective row. The significance is determined by a two-sided paired Wilcoxon-Mann-Whitney test for these two distributions with $n = 10$

		Moses		
Training	Testing	fastText	MLP	Sig.
translated	native	0.122(±0.013)	0.203(±0.019)	***
native	translated	0.084(±0.013)	0.092(±0.017)	ns
		Google Cloud Translation API		
Training	Testing	fastText	MLP	Sig.
translated	native	0.036(±0.006)	0.091(±0.013)	***
native	translated	0.095(±0.012)	0.127(±0.012)	***

TABLE 16
Augmenting English Reviews: Results (Micro-Averaged F_1 Score)

	fastText	MLP
Moses	0.896(±0.011)	0.884(±0.013)
Yandex Translation API	0.899(±0.009)	0.885(±0.009)

TABLE 17
Augmenting German Reviews: Results (Micro-Averaged F_1 Score)

	fastText	MLP
Moses	0.899(±0.011)	0.868(±0.008)
Google Cloud Translation API	0.908(±0.011)	0.885(±0.010)

The transfer error resulting from using translated reviews instead of native lingual ones for either training or validation, is shown in Tables 14 and 15. The significance of the discrepancy between the distribution of fastText and the distribution of the MLP is given in the column" Sig." where "ns" means no significance ($p > 0.05$), * means $p \leq 0.05$, ** means $p \leq 0.01$, and *** means $p \leq 0.001$. The significance is determined using a two-sided paired Wilcoxon-Mann-Whitney test [48].

4.2 Augmenting Datasets: Results

Tables 16 and 17 show the results of the experiments of the dataset augmentation, for both source languages: German and English as well as for the two classification algorithms and the different translation algorithms.

5 Discussion

In the following, the results of the experiments described in the preceding sections will be discussed.

5.1 Influence of Translations

The best classification results are achieved in cases where a classifier is applied to a dataset, that originates from the same probability distribution as the dataset the classifier was trained on. This holds true for both, the native lingual data sets as well as for the translated datasets. A fastText classifier that was trained on native lingual German reviews with fastText achieves a micro averaged F_1 score of 0.917

TABLE 18
Difference Between Moses and the Respective Commercial API for Particular Experiments

	train: transl., test: native	train: native, test: transl.
German, fastText	0.087	0.008
German, MLP	0.111	−0.018
English, fastText	0.005	0.005
English, MLP	0.006	0.007

with a standard deviation of 0.008. The same classifier, applied to a translated version of the dataset achieves a score of $0.909(\pm 0.007)$ in case of the Yandex Translation API and it achieves a score of $0.902(\pm 0.007)$ in case of the Moses statistical classifier.
In case of English being the native language, the micro-averaged F_1 scores of the classifiers performing on Moses translations are respectively 0.007 points below the scores of the Yandex translations. In case of German being the native language, this discrepancy is higher. The scores of the classifiers performing on the Moses reviews are each more than 0.039 below the scores of the other classifiers. There are multiple possible explanations for this result. First, the Yandex translations could be worse than the Google translations in terms of classification performance. Therefore, the results of the classifier performing on the Yandex reviews are closer to the (always worse) results of the classifier performing on Moses translations. Based on the fact, that the Google translator works with a different technology (NMT) compared to the Yandex and the Moses translator (SMT), this is not unlikely.
Second, the reason for the deviating discrepancy could lie in the Moses translations. The results of the classifiers working on native English reviews are worse than the results of the classifiers working on native German reviews[12]. Hence, it is not straightforward that the classification on Google or Yandex translations would achieve the same score as they did in the other language. The Yandex translations could be on par with the Google translations even though they differ in another language. If this is the case, the discrepancy could result from Moses performing worse on English to German than on German to English translations.
To verify which of these assumptions hold true, the Yandex and the Google translator would each have to be applied to both languages.

The translations of the Moses translator seem to be poor to a human reader, however, they provide reasonable results compared to commercial translation APIs, in particular, the Yandex Translation API and the Google Cloud Translation API. However, there are qualitative differences. In cases where a classifier is trained on native lingual reviews and applied to translated reviews, the discrepancy between Moses and the commercial APIs (see Table 18) is relatively small. In case of German, where Google was the commercial API, the difference between (1) training on translated reviews and testing on native, and (2) training on native and testing on translated is very large. In case of Yandex being the commercial API, there is almost

12. In case of fastText they are significantly (*) worse.

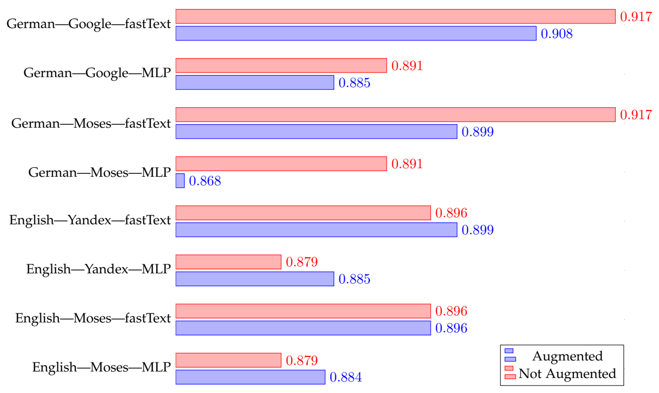

Fig. 3. Results of the Data Augmentation

no difference. Again, this could have the same reasons as described in the previous paragraph.

Using translated reviews for training and native lingual reviews for testing or using native lingual reviews for training and translated reviews for testing results in a drop of classification performance compared to sample training and validation set from the same distribution. In all cases, this transfer error is higher for the MLP classifier compared to the fastText classifier. In 7 out of 8 cases, the discrepancy between these two distributions is highly significant. Thus, the fastText classifier, in general, is less prone to errors resulting from using translated texts instead of native lingual texts than the multilayer perceptron using BOW features. Based on this discovery, the question is whether word embeddings *in general* are more stable in terms of the translation loss than BOW representations. To verify or falsify this assumption, the described experiments would have to be performed on a larger scale. Different word embedding techniques would have to be compared to different BOW representations, respectively working with different classifiers. Moreover, fastText incorporates character-level n-grams as well as word level n-grams. Further research should examine whether the superiority of fastText is based on these characteristics or whether word embedding techniques in general, including word2vec, are better than BOW based approaches.

5.2 Augmenting Datasets

The results of the classifiers, whose training set was augmented with the translated corpus are shown in Tables 16 and 17. The results are compared with the results of the classifiers trained only on the native lingual reviews in Figure 3. There is no evidence, that augmenting a classifier with translated data helps improving its micro-averaged F_1 score. Instead, in cases where a German classifier is augmented with translated Moses reviews, the classification performance drops significantly ($p < 0.05$). In all other cases, there is no significant difference between the results of the classifier trained with the augmented dataset and the results of the classifier only trained on the native lingual reviews.

6 Conclusion

The previous experiments have shown the advantages of using fastText instead of a multilayer perceptron for translation-based multilingual text classification. However, since these results are based on only two algorithms, it is not possible to conclude that word embeddings in general are more stable regarding translation loss. For making more general assumptions, additional word-embedding approaches should be compared to other BOW approaches.

The experiments have also shown that classifiers trained on Moses-translated reviews perform worse than classifiers trained on Yandex- or Google-translated reviews. Even though, the discrepancy between the Moses-Yandex translations is less than between the Moses-Google translations, no superiority of one algorithm (Google or Yandex) over the other concludes.

The experiments on data augmentation brought no proof, that a text classifier can be enhanced by using more training data, that was translated from other languages. However, there is no implication that a text classifier *cannot* be enhanced by using for example a larger corpus consisting of such data.

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[3] T. Mikolov and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[4] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, vol. 5, 2005, pp. 246–252.

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[6] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[7] H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan, "A system for real-time twitter sentiment analysis of 2012 u.s. presidential election cycle," in *Proceedings of the ACL 2012 System Demonstrations*. Association for Computational Linguistics, 2012, pp. 115–120.

[8] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, vol. 2006, pp. 120–128, 2006.

[9] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification," *Proceedings of the Association for Computational Linguistics (ACL)*, pp. 478–486, 2007.

[10] H. Daumé III, "Frustratingly easy domain adaptation," *arXiv preprint arXiv:0907.1815*, 2009.

[11] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 513–520.

[12] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[13] D. Bollegala, D. Weir, and J. Carroll, "Cross-domain sentiment classification using a sentiment sensitive thesaurus," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 8, pp. 1719–1731, 2013.

[14] G. Zhou, Z. Xie, J. X. Huang, and T. He, "Bi-transferring deep neural networks for domain adaptation," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 322–332, 2016.

[15] B. Liu, *Sentiment analysis: Mining opinions, sentiments, and emotions*. New York, NY: Cambridge University Press, 2015.

[16] R. Socher, C. C.-Y. Lin, A. Y. Ng, and C. D. Manning, "Parsing natural scenes and natural language with recursive neural networks," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. Omnipress, 2011, pp. 129–136.

[17] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," *Proceedings of the ACL - 02 conference on Empirical methods in natural language processing-Volume 10*, vol. 2002, pp. 79–86, 2002.

[18] Y. Kim, "Convolutional neural networks for sentence classification," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, 2014.

[19] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Toutanova and H. Wu, Eds. Association for Computational Linguistics, 2014, pp. 655–665.

[20] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, 2015, pp. 649–657.

[21] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," in *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Languages Resources Association (ELRA), 2010.

[22] A. Tumasjan, T. Sprenger, P. Sandner, and I. Welpe, "Predicting elections with twitter: What 140 characters reveal about political sentiment," *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pp. 178–185, 2010.

[23] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science*, vol. 2, no. 1, pp. 1–8, 2011.

[24] A. Sarlan, C. Nadam, and S. Basri, "Twitter sentiment analysis," in *Proceedings of the 6th International Conference on Information Technology and Multimedia*. IEEE, 2014, pp. 212–216.

[25] A. Dalmia, M. Gupta, and V. Varma, "Iiit-h at semeval 2015: Twitter sentiment analysis – the good, the bad and the neutral!" in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, 2015, pp. 520–526.

[26] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 385.

[27] O. Irsoy and C. Cardie, "Opinion mining with deep recurrent neural networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Q. C. R. I. Alessandro Moschitti, G. Bo Pang, and U. o. A. Walter Daelemans, Eds. Association for Computational Linguistics, 2014, pp. 720–728.

[28] X. Wang, Y. Liu, C. SUN, B. Wang, and X. Wang, "Predicting polarities of tweets by composing word embeddings with long short-term memory," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong and M. Strube, Eds. Association for Computational Linguistics, 2015, pp. 1343–1353.

[29] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, K. Knight, A. Nenkova, and O. Rambow, Eds. Association for Computational Linguistics, 2016, pp. 1480–1489.

[30] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," *arXiv preprint arXiv:1309.4168*, 2013.

[31] S. Lauly, A. Boulanger, and H. Larochelle, "Learning multilingual word representations using a bag-of-words autoencoder," *arXiv preprint arXiv:1401.1803*, 2014.

[32] K. Spärck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[33] P. Koehn, "Europarl: A parallel corpus for statistical machine translation," in *Conference Proceedings: the tenth Machine Translation Summit*, vol. 5, 2005, pp. 79–86.

[34] P. Prettenhofer and B. Stein, "Cross-language text classification using structural correspondence learning," in *Proceedings of the 48th annual meeting of the association for computational linguistics*, 2010, pp. 1118–1127.

[35] C. Banea, R. Mihalcea, J. Wiebe, and S. Hassan, "Multilingual subjectivity analysis using machine translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008, pp. 127–135.

[36] H. Zhou, L. Chen, F. Shi, and D. Huang, "Learning bilingual sentiment word embeddings for cross-language sentiment classification," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong and M. Strube, Eds. Association for Computational Linguistics, 2015, pp. 430–440.

[37] X. Zhou, X. Wan, and J. Xiao, "Cross-lingual sentiment classification with bilingual document representation learning," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Erk and N. A. Smith, Eds. Association for Computational Linguistics, 2016, pp. 1403–1412.

[38] J. Wehrmann, W. Becker, H. E. L. Cagnini, and R. C. Barros, "A character-based convolutional neural network for language-agnostic twitter sentiment analysis," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 2384–2391.

[39] J. McAuley, R. Pandey, and J. Leskovec, "Inferring networks of substitutable and complementary products," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 785–794.

[40] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens *et al.*, "Moses: Open source toolkit for statistical machine translation," in *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, 2007, pp. 177–180.

[41] P. Koehn, *Statistical machine translation*, 5th ed. Cambridge: Cambridge University Press, 2014.

[42] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[43] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado *et al.*, "Google's multilingual neural machine translation system: Enabling zero-shot translation," *arXiv preprint arXiv:1611.04558*, 2016.

[44] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, pp. 1724–1734. [Online]. Available: http://www.aclweb.org/anthology/D14-1179

[45] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc, 2014, pp. 3104–3112.

[46] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[48] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

Appendix A
Share of Common Words

As introduced in section 1, domain adaptation systems try to alter a classifier that was trained in the source domain in a way, that this classifier shows better performance in the source domain than it did before the domain adaptation. In the case of text classification tasks, two domains share a certain amount of words while other words only occur in specific domains. Previous experiments indicate that a larger amount of common words yields a smaller drop of classification performance.

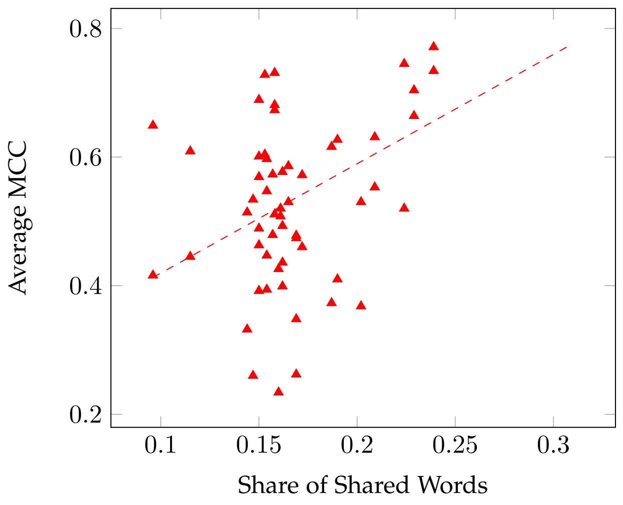

Fig. 4. Amount of Common Words and Average MCC: the regression line is computed using quantile regression with $\tau = 0.5$

Figure 4 shows the average Matthews Correlation Coefficient (MCC) of a model trained in a source domain and applied to a (different) target domain and the according share of common words. As figure 4 suggests, there is a weak correlation of 0.462 between these two variables. If a larger share of common words yields a smaller drop in classification performance, it would be worthwhile to increase this share.

```
1: define a threshold $\epsilon$
2: create word vectors
3: $\vec{v}_s = \{w_1^s, w_2^s, w_{n_s}^s\}$ is the source vocabulary with terms $w_1^s, w_2^s, \ldots, w_{n_s}^s$ and $\vec{v}_t = \{w_1^t, w_2^t, w_{n_t}^t\}$ is the target vocabulary with terms $w_1^t, w_2^t, \ldots, w_{n_t}^t$
4: for every $w_i^t \in \vec{v}_t$ do
5:     if $\exists w_j^s$ with $sim(w_j^s, w_i^t) < \epsilon$ then
6:         find $k$ : $\min_{\forall k \in \{1,2,\ldots,n_s\}} sim(w_k^s, w_i^t)$
7:         replace $w_i^t$ in $\vec{v}_t$ with $w_k^s$
8:     end if
9: end for
10: perform domain adaptation with source vocabulary $\vec{v}_s$ and target vocabulary $\vec{v}_t$
```

Fig. 5. Unknown Word Replacement

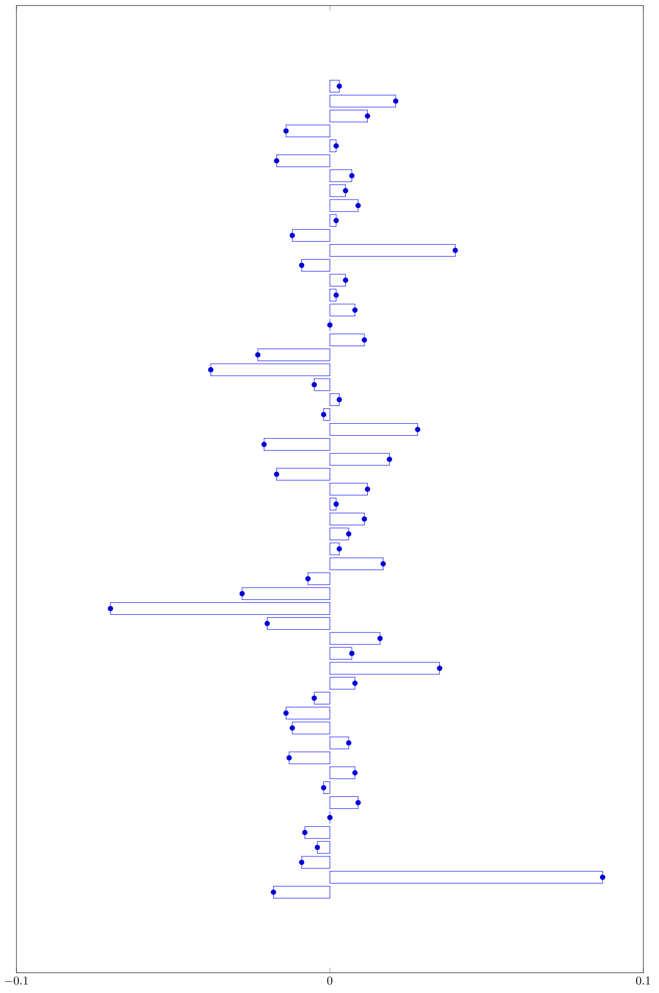

Fig. 6. Difference between the Classification with unknown word replacement and without unknown word replacement. The differences almost sum up to 0.

The algorithm presented in Figure 5 shows one assumed possible way to increase this share. It replaces words in the target domain, that did not occur in the source domain with words from the source domain, that are similar to the original words. The idea is, that two deviating domains use slightly different terms that have a similar meaning. If a word in the target domain did not occur in the source domain, a classifier was not able to consider them as a feature. The similarity of two terms can be computed using word embeddings and the cosine distance. If the source and the target vocabulary are considered as fixed, the only hyperparameter is the threshold ϵ. However, this hyperparameter is hard to determine.

Figure 6 shows the results of the algorithm ($\epsilon = 5.0$) in comparison to a classifier that left out this preprocessing step. Although some values differ to a great degree, the mean difference sums up to approximately zero.

A two-sided paired Wilcoxon-Mann-Whitney-Test [48] yields a p-value of 0.918. Hence, the result is not significant.

Fasttext and Gradient Boosted Trees at GermEval-2017 Tasks A and B on Relevance Classification and Document-level Polarity

Abstract

This paper describes the submissions to the *Shared Task on Aspect-based Sentiment in Social Media Customer Feedback* for the *GermEval 2017*-workshop for the two subtasks *Relevance Classification* (task A) and *Document-level Polarity* (task B). For each subtask, the results of the same three systems were submitted: a fastText classifier, enhanced with pretrained vectors, gradient boosted trees (GBTs) trained on bag-of-words (BOWs), and an ensemble of GBTs, respectively trained on word embeddings and on BOWs. For the subtask "Relevance Classification", the best system yields a micro-averaged F_1 score of 0.895 on the dev set and 0.907 using 5-fold cross validation on the train and the dev set. For the subtask "Document-level Polarity", the best system achieves 0.782 on the test set and 0.775 using 5-fold cross validation on the train and the dev set. The proposed system achieved the second place of twelve systems submitted by seven teams for task A for both test sets. For task B, the proposed system achieved the first place for test set one and the second place for test set two of 17 systems submitted by 8 different teams.

1 Introduction

For companies, customer feedback in social networks is a valuable resource for improving the own service. Often, customers propose improvements and show points of criticism companies were unaware of. Moreover, it is important to get an impression of the opinions customers hold with regard to the own company. Manually separating the relevant feedback from the irrelevant, however, needs many human resource and is therefore expensive. The *GermEval 2017 Shared Task on Aspect-based Sentiment in Social Media Customer Feedback* workshop (Wojatzki et al., 2017) addresses the automatic processing of German-language customer feedback regarding its different characteristics. Therefor, four different subtasks were defined: binary classification of whether a feedback is relevant to a given instance (e.g. a company) (task A), categorization of the feedback into sentiment classes (positive, neutral, and negative) (task B), binary classification of a specific aspect into sentiment classes (positive and negative) (task C), and opinion target extraction (task D). In order to elaborate the different subtasks, a training set of customer feedbacks regarding the German railroad company "Deutsche Bahn" was provided. Together with every feedback text, class labels for each subtask were given. The goal was to develop a classifier with a high micro-averaged F_1 score on the prediction of those class labels.
Word embeddings trained with the objective to predict sentiment polarity were successfully applied at SemEval 2014 workshop (Tang et al., 2014a; Tang et al., 2014b). Ensembles of convolutional neural networks (CNNs) and long short-term memories (LSTMs), trained on pre-trained word embeddings achieved state-of-the-art performance at SemEval 2017 workshop (Cliche, 2017). Zhang et al. presented character-level CNNs, trained on one-hot encoded character vectors for text classification tasks (Zhang et al., 2015). GBTs (Friedman, 2001) have shown good results in a variety of classifications tasks. 17 out of 29 systems that have been published on Kaggle's blog during 2015 used XGBoost, a framework that implements GBTs (Chen and Guestrin, 2016).
The following sections describe the work on two out of the four given subtasks ("Relevance Classification" and "Document-level Polarity").

2 Dataset

The dataset for the classification task consists of customer feedback texts collected from various social media platforms, the hyperlink to the resource of each text, and the annotations of class labels per subtask. The class labels contained information about the respective document-level sentiment polarity (positive, neutral, or negative), the binary relevance label (denoting whether the text contains feedback about the "Deutsche Bahn"), as well as annotations for other subtasks which will not be considered in this paper. The data set was split into a training set (train) and a training test set (dev). Additionally, a test set (test) without class labels was provided at a later stage.

Dataset	# Reviews	Relevance	
		# true	# false
train	19449	16217	1937
dev	2375	3232	438
test	4408	-	

Table 1: Number of Reviews in the Data Sets and their Respective Relevance Classes

Dataset	# Reviews	Document-level Polarity		
		# positive	# neutral	# negative
train	19449	1179	13222	5048
dev	2375	149	1637	589
test	4408		-	

Table 2: Number of Reviews in the Different Document-level Polarity Classes

The distribution of the feedbacks over the different classes are shown in Tables 1 and 2. The classes are not equally distributed, neither in the case of relevance classification nor in the case of document-level sentiment polarity classification. The average lengths in the different relevance and document-level polarity classes are shown in Tables 3 and 4.

3 Text Preprocessing

Out of the whole dataset, only the review texts were used as a feature. The hyperlinks provided in the dataset were not considered. Each review text was tokenized to extract single terms using whitespaces, percents signs, forward slashes, and plus signs as delimiters.
A large amount of the data originates from Tweets. Therefore, special attention was paid to Twitter-specific text preprocessing. Frequently occurring Twitter usernames related to the "Deutsche Bahn" like "@DB_Bahn", "@Bahnansagen", or "@DB_Info" are pooled by replacing them with "⟨⟨⟨db_username⟩⟩⟩". Other terms containing an "@" are replaced with the token "⟨⟨⟨twitter_username⟩⟩⟩"
The terms "S-Bahn" and "S Bahn" are replaced with "sbahn".
The emoticons ":-)", ":)", and ":-))" are replaced by the token "⟨⟨⟨happy_smiley⟩⟩⟩" . The emoticons ":-D" and "xD" are replaced by "⟨⟨⟨laughin_smiley⟩⟩⟩". The emoticons ":-(" and "":(" are replaced by "⟨⟨⟨sad_smiley⟩⟩⟩".

Dataset	Relevance	
	true	false
train	29.761	42.231
σ	25.164	24.943
dev	29.093	41.374
σ	25.228	25.507
test	69.373	
σ	231.134	

Table 3: Average Word Count in Different Relevance Classes and the Corresponding Standard Deviation

Dataset	Document-level Polarity		
	positive	neutral	negative
train	28.355	32.597	30.647
σ	22.477	25.599	25.992
dev	28.268	32.283	29.569
σ	19.768	26.134	25.769
test		69.373	
σ		231.134	

Table 4: Average Word Count in Different Document-level Polarity Classes and the Corresponding Standard Deviation

Punctuation characters are removed. An exception is three or more repetitions of question marks, exclamation points, and periods. These are replaced with the tokens "⟨⟨⟨strong_question⟩⟩⟩", "⟨⟨⟨strong_exclamation⟩⟩⟩" and "⟨⟨⟨annoyed_dots⟩⟩⟩" in order to retain the emotion, expressed by the usage of such a writing style.
Many of the feedbacks contain time specifications, for example in the following tweet: "Nach 25 Minuten ist mein Nebenschwitzer in der Bahn ausgestiegen". These time specifications are replaced by "⟨⟨⟨time⟩⟩⟩" using a regular expression. Money amounts are replaced by "⟨⟨⟨money⟩⟩⟩". Other numbers are replaced by "⟨⟨⟨number⟩⟩⟩". Furthermore, hyperlinks occurring within the text are also replaced by the token "⟨⟨⟨hyperlink⟩⟩⟩" in order to prevent overfitting. Finally, quotation marks are replaced by "⟨⟨⟨quotation⟩⟩⟩".
After the grouping of related terms, the remaining text is transformed to lower case and subsequently stemmed using the German stemmer in Snowball[1]. The stemmer also replaces special German characters. The characters *ß* is replaced by *ss* and

[1]available from: http://snowballstem.org/, last access: 07/19/2017, license: 3-clause BSD

äöü are replaced by *aou*.
In a next step, the feedbacks are vectorized using the hashing trick (Weinberger et al., 2009). For reasons of computational efficacy, the resulting vectors are condensed to length 16,384 applying a modulo operation. The vector entries are the TF-IDF (term frequency - inverse document frequency) values of the terms (Spärck Jones, 1972).

4 Additional Features

The TF-IDF vectors are enriched with additional information from three additional feature sources: LIWC-features, word defectiveness, and sentiment lexicon.
The LIWC-features are determined using the German version of the LIWC (Linguistic Inquiry and Word Count) computer program (Tausczik and Pennebaker, 2010; Wolf et al., 2008), that "counts words in psychological meaningful categories" (Tausczik and Pennebaker, 2010). The LIWC tool outputs 93 decimal values, that can directly be used as features.
In addition to the LIWC-features, the vector is augmented with a feature expressing the belonging of a review to a certain cluster of reviews. To generate these features, the vector space of the fastText word representations was clustered into 100 clusters. In order to determine the cluster centers, a large vector space model, trained on a snapshot of all German Wikipedia articles, a monolingual news corpus [2], and the feedback texts themselves (train+dev+test). The *k*-means algorithm was trained on this vector space with the objective to determine 100 cluster centers.
Furthermore, binary features resulting from SentiWS (Goldhahn et al., 2012) were used. Therefor, a vector with the length of positive and negative words was created. Each index in this vector is connected to one word in SentiWS. At transformation time, the respective cell in this vector is set to 1 if the word is contained in the review and to 0 otherwise. The list of words considered as positive has length 17,626 and the list of negative words has length 19,961, resulting in 37,587 additional features.
Finally, a feature expressing word defectiveness was created. For this purpose, the German version of the LanguageTool[3] was used. The LanguageTool is able to detect a variety of faulty language, including grammatical errors, missing punctuation or wrong capitalization. However, since the lower-case transformed and punctuation marks freed version of the text was provided to it, only errors that match the *GermanSpellerRule* were considered, which detects spelling errors in German language. The feature was created by dividing the number of times the *GermanSpellerRule* matches in a feedback by the number of words in the respective feedback.

5 Feature Selection

The 1000 top features are chosen out of the 16,384 hashed BOW vectors and the additional features except for the SentiWS-features. The features were selected performing a χ^2-test (Liu and Setiono, 1995, pp. 36,37). Tables 5 and 6 show the top 20 features for the sentiment class and the relevance class, applying χ^2-feature selection and mutual-information feature selection to the BOW features. The additional features were not considered for the creation of these tables. Since using the hashing trick causes hash collisions, the top features were determined without using the hashing track and may therefore vary slightly from the actual features.
The mutual information analysis results in many stop words like articles or pronouns whereas the χ^2 analysis yields more meaningful terms like "streik" (strike) or "verspat", which is the stem of "Verspätung" (delay).

5.1 Classifiers

Three different systems were developed and tested, which will be explained in the following.
The first system are GBTs (Friedman, 2002) on BOW vectors (FHDO_GBT_BOW). For this system, the hashed TF-IDF weights of individual terms are merged with the LIWC-features and provided to the feature selection algorithm. The feature selection yields the top 1000 features. For the document-level sentiment polarity classification, the top 1000 features include six LIWC features: *body, netspeak, other punctuation, positive emotion, auxiliary verbs,* and *comparisons*. For relevance classifications, no LIWC features are among the top 1000 features. GBTs are trained on these

[2]http://www.statmt.org/wmt14/translation-task.html, last access: 08/15/2017

[3]available from https://languagetool.org, last access: 07/19/2017, license: LGPL 2.1

χ^2 (relevance)	χ^2 (document-level polarity)
gestartet	franzos
kehrt	streikend
asteroid	notrufsyst
germanwing	schnell
schweiz	aufatm
barbi	grenzuberschreit
weltmeist	bahnstreik
stellenangebot	lokfuhr
bahn	storung
job	sncf
cavendish	regionaldirektion
lik	tarifkonflikt
osterreich	schlichtung
⟨⟨⟨db_username⟩⟩⟩	funkloch
ad	beend
anna	paris
schwebebahn	verspat
ors	gdl
bigg	streik
lufthansa	beendet

Table 5: Top Features Using χ^2-Selection

MI (relevance)	MI (document-level polarity)
es	fur
nicht	den
zu	im
re	es
im	auf
von	ich
den	re
fur	nicht
ist	das
auf	mit
das	ist
mit	⟨⟨⟨db_username⟩⟩⟩
⟨⟨⟨hyperlink⟩⟩⟩	ein
ein	in
in	⟨⟨⟨hyperlink⟩⟩⟩
und	⟨⟨⟨number⟩⟩⟩
⟨⟨⟨number⟩⟩⟩	und
der	die
die	der
bahn	bahn

Table 6: Top Features Using Mutual Information-Selection

1000 top features. The maximal depth of the trees was set to 10, the number of iterations to 30. For document-level sentiment polarity classification, a one-vs-rest strategy was used since the implementation of this system only supports GBTs for binary classification. The gini-coefficient is used for impurity calculation and logistic loss as the loss function. The step size was initialized with 0.1.
The second system is a fastText classifier (Joulin et al., 2016), trained on the preprocessed text (FHDO_FT). The word stemming was omitted for fastText classification. Generally, the default configuration was used. The dimensionality of the word vectors was set to 100, the size of the context window was set to five, negative sampling was used for loss computation and the learning rate was initialized with 0.05. Furthermore, the large unsupervised corpus described in the previous paragraph was used as an additional feature.
The supervised fastText algorithm is constructed as follows. Instead of predicting a word, the goal is to predict a class (or label) - for example, *true* or *false* in the sense of document relevance. Similar to the CBOW model (Mikolov et al., 2013), fastText works with (language-dependent) word embeddings. In addition to the CBOW model, fastText word embeddings also make use of character-level n-grams. In order to represent a document, word embeddings are averaged and word order is discarded. The model is capable of handling sentences with a varying number of words. fastText also uses bigram features in order to retain some information about the word order. The use of bigrams is based on the impact of bigrams on classification accuracy in sentiment analysis (Wang et al., 2012). The model is trained by minimizing the negative log-likelihood over classes

$$-\frac{1}{N} \cdot \sum_{n=1}^{N} y_n \cdot \log(f(B \cdot A \cdot x_n)) \qquad (1)$$

where x_n "is the normalized bag of features of the n-th document" (Joulin et al., 2016), y_n the label, and A and B are weight matrices.

The third model is an ensemble of two GBT classifiers, where one classifier is trained on the BOW vectors and the other on the word embedding vectors, merged with the one-hot encoded cluster belonging and the LIWC features (FHDO_GBT_NSMBL).

System	5-fold CV	Train / Dev
	Document-level Polarity	
Baseline	0.700(±0.008)	0.710
FHDO_FT	0.775(±0.007)	0.782
FHDO_GBT_BOW	0.728(±0.006)	0.729
FHDO_GBT_NSMBL	0.751(±0.004)	0.754
FT_UNPROCESSED	0.757(±0.006)	0.760
FT_NO_PRETRAIN	0.756(±0.006)	0.764
GBT_TOP_1000	0.728(±0.007)	0.728
GBT_LT_TOP_1000	0.728(±0.007)	0.728
GBT_W2V_ONLY	0.720(±0.009)	0.727
GBT_W2V_LIWC	0.718(±0.008)	0.725
GBT_W2V_SENLEX	0.734(±0.002)	0.731
MLP_TOP_1000	0.734(±0.005)	0.744
MLP_W2V_ONLY	0.719(±0.011)	0.721
MLP_W2V_LIWC	0.585(±0.008)	0.587
MLP_W2V_LIWC_SENLEX	0.646(±0.007)	0.653
MLP_W2V_SENLEX	0.731(±0.006)	0.742
	Relevance	
Baseline	0.881(±0.010)	0.882
FHDO_FT	0.907(±0.007)	0.895
FHDO_GBT_BOW	0.893(±0.006)	0.878
FHDO_GBT_NSMBL	0.887(±0.004)	0.866
FT_UNPROCESSED	0.891(±0.006)	0.896
FT_NO_PRETRAIN	0.900(±0.005)	0.894
GBT_TOP_1000	0.885(±0.005)	0.878
GBT_LT_TOP_1000	0.886(±0.006)	0.878
GBT_W2V_ONLY	0.862(±0.007)	0.852
GBT_W2V_LIWC	0.872(±0.008)	0.858
GBT_W2V_SENLEX	0.862(±0.004)	0.847
MLP_TOP_1000	0.864(±0.002)	0.858
MLP_W2V_ONLY	0.872(±0.005)	0.872
MLP_W2V_LIWC	0.107(±0.005)	0.106
MLP_W2V_LIWC_SENLEX	0.810(±0.006)	0.796
MLP_W2V_SENLEX	0.870(±0.005)	0.858

Table 7: Results of Submitted Systems and Baseline (micro-averaged F_1 score)

The baseline results in Table 7 are provided by organizers of the GermEval 2017 workshop [4]. It is an SVM using term frequency and a sentiment lexicon as features.

5.2 Results

Table 7 shows the results of all three systems. The results were computed using 5-fold cross-validation on both the train and the dev set. The final submission was created by training on both sets and applying the trained model on the two test sets. The first column shows the average of the 5-fold cross-validation and the respective standard deviation while the second column gives the results of the classifier trained on the given training set and validated on the given dev set.

The models whose names start with "FHDO" are the submitted models, described in the previous section. The other models were developed for comparison reasons.

FT_UNPROCESSED is a model applying the same fastText classifier used for FHDO_FT to an unprocessed version of the dataset, where only the tokens were split using whitespaces as delimiters.

FT_NO_PRETRAIN is trained as FHDO_FT, preprocessing the text, but without incorporating the pre-trained vectors.

GBT_TOP_1000 are GBTs with the same configuration as the ones in the submissions, that has been trained on the 1000 top TF-IDF features from the feature selection only.

GBT_W2V_ONLY are GBTs trained on the fastText word embeddings only, whereby the word embeddings were again computed using the pre-trained vectors.

GBT_W2V_LIWC is the same setup, but taking into account the LIWC-features as additional features.

Accordingly, GBT_W2V_SENLEX are the same setup as GBT_W2V_ONLY but considering the sentiment lexicon features as additional features.

In addition the the GBT classifier, a feedforward multilayer perceptron (MLP) with one hidden layer with 500 hidden units was used. The output layer was trained using the softmax function while the hidden layer uses the sigmoid function. The number of output units varies depending on the number of classes. For the MLP, the same experiments were performed as for the GBT. Consequently, the models starting with MLP_ have the same setup like the GBT_-models. However, no one-vs-rest strategy was used for the MLP.

5.3 Conclusion

The best results for both subtasks have been achieved applying the fastText classifier with the pre-trained vectors to the preprocessed dataset. The micro-averaged F_1 score decreased by approximately two points when performing the fastText classification without preprocessing the dataset. Still, the result was better than the other tested classifiers. Using fastText without the pre-trained vectors causes a drop of about two points in case of document-level sentiment polarity classification and of 0.7 points in case of relevance classification. Applying GBT classifiers to BOW representations

[4] https://sites.google.com/view/germeval2017-absa/ , last access: 07/24/2017

instead of fastText word embedding representations resultes in a higher micro-averaged F_1 score for both subtasks.
Using an ensemble of two different GBT classifiers on two different variants of the dataset yields a model with low variance for both subtasks.

References

Tianqi Chen and Carlos Guestrin. 2016. Xgboost. In Balaji Krishnapuram, Mohak Shah, Alex Smola, Charu Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pages 785–794. ACM Press.

Mathieu Cliche. 2017. BB_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. *arXiv preprint arXiv:1704.06125*.

Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Jerome H. Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378.

Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pages 759–765.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Huan Liu and Rudy Setiono. 1995. Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, pages 388–391.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Robert Remus, Uwe Quasthoff, and Gerhard Heyer. 2010. Sentiws-a publicly available german-language resource for sentiment analysis. In *LREC*.

Karen Spärck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. 2014a. Coooolll: A deep learning system for twitter sentiment classification. *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 208–212.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014b. Learning sentiment-specific word embedding for twitter sentiment classification. In Kristina Toutanova and Hua Wu, editors, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565. Association for Computational Linguistics.

Yla R. Tausczik and James W. Pennebaker. 2010. The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54.

Hao Wang, Dogan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan. 2012. A system for real-time twitter sentiment analysis of 2012 u.s. presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*, pages 115–120. Association for Computational Linguistics.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120.

Michael Wojatzki, Eugen Ruppert, Sarah Holschneider, Torsten Zesch, and Chris Biemann. 2017. Germeval 2017: Shared task on aspect-based sentiment in social media customer feedback. In *Proceedings of the GSCL GermEval Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*.

Markus Wolf, Andrea B. Horn, Matthias R. Mehl, Severin Haug, James W. Pennebaker, and Hans Kordy. 2008. Computergestützte quantitative textanalyse: Äquivalenz und robustheit der deutschen version des linguistic inquiry and word count. *Diagnostica*, 54(2):85–98.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Appendices

A Data Augmentation

In the following, the possibilities to enhance a text classifier by incorporating additional data sources are discussed.

A.1 Introduction

Data augmentation is a frequently used technique in image classification. Generating additional training data by for example mirroring, rotating, scaling images helps by making classifiers invariant to certain transformation and more accurate. These techniques are not applicable in text classification settings. However, there are various additional expert-designed resources for language processing, that might help improving the performance of a text classifier. Two different resources and their ability to increase the performance of a sentiment classification system are determined in this section.

A.2 Experimental Setup

For the general classification task, the native lingual German amazon product review corpus[5] was used. Reviews with more than three stars were considered as positive whereas reviews with less than three stars were considered as negative. Neutral reviews were ignored. For the enhancement of the classifier, two different data source were used: the *Linguistic Inquiry and Word Count* computer program and SentiWS, which are described in the following.

Linguistic Inquiry and Word Count (LIWC)[6] (Tausczik and Pennebaker, 2010) is one of these additional resources. This dictionary-based text analysis program assigns words to certain syntactical and psychological categories and yields the relative frequency of every category. Higher level categories are[7] "Summary Dimensions" (e.g. total word count, words per sentence), "Punctuation Marks" (e.g. periods, comma, exclamation marks), "Function Words" (e.g. pronouns, articles), "Other Grammar" (e.g. verbs, adjectives), "Affect" (e.g. positive emotions, negative emotions), "Social" (e.g. family, friends), "Cognitive Processes" (e.g. cause, discrepancies), "Perceptual Processes (hear, see, feel), "Biological Processes" (e.g. body, health), "Drives" (e.g. power, risk), "Time Orientation" (past / present / future focus), "Relativity" (motion, space, time), "Personal Concerns" (e.g. work, religion), and "Informal Language" (e.g. swear, netspeak). There are multiple dictionaries for other languages, especially for German language. The advantage of LIWC is that it yields new features that can be concatenated with an existing feature vector.

SentiWS[8] (Remus et al., 2010) is a resource for German sentiment analysis, consisting of lists of words with their respective document-level sentiment polarity. Each word is listed with its part-of-speech tag, its polarity weight, which is positive in case of positive sentiment and negative in case of negative sentiment, and a list of its inflections. There are various possible ways of incorporating this resource for data augmentation.

The text was preprocessed as described in section 3: hashed BOW vectors with TF-IDF weights were created. No feature selection was performed.

A logistic regression model was used for the classification task. Four different experiments were distinguished. Each of them was performed using 10-fold cross validation.
First, the classifier was trained on the text only without incorporating any additional data sources. This is model is referred to as *LR_TEXT_ONLY*.
Second, a model augmented with SentiWS was evaluated. It is referred to as *LR+SENLEX*. The additional SentiWS features, were produced as described in section 4. A binary value in the additional vector indicates the existence of the respective LIWC term in the particular document.
Third, a model augmented with the LIWC features was created. Therefore, the particular document was processed using LIWC and the resulting features were appended to the feature vector. This model is referred to as *LR+LIWC*.
The last model incorporates both, the LIWC and the SentiWS features. The feature vector was sim-

[5]See the other document: *Sentiment Analysis Based on Word Embeddings: Possible Improvements and Transfer to German Language*

[6]http://liwc.wpengine.com, last access: 08/11/2017, license: academic or commercial

[7]In version 1.3.1 with the German LIWC 2015 dictionary

[8]http://wortschatz.uni-leipzig.de/de/download, last access: 08/11/2017, license: CC BY-NC-SA 3.0, current version: v1.8c

Model	∅micro-averagedF_1	Significance
LR_TEXT_ONLY	0.806(±0.014)	
LR+SENLEX	0.829(±0.014)	*
LR+LIWC	0.817(±0.016)	*
LR+SENLEX+LIWC	0.844(±0.007)	*

Table 8: Results of the Data Augmentation Experiments: the significance is always determined with respect to LR_TEXT.

ply created by merging the hashed BOW features, the LIWC features, and the SentiWS features. This model is referred to as *LR+SENLEX+LIWC*.

A.3 Results

The results of the experiments are shown in Table 8. The different systems were evaluated using the micro-averaged F_1-score. Each model is denoted with its respective standard deviation given in braces behind the scores.

The significance of each model is indicated using the following convention: "ns" means no significance ($p > 0.05$), * means $p \leq 0.05$, ** means $p \leq 0.01$, and *** means $p \leq 0.001$. The significance is evaluated using a two-sided paired Wilcoxon-Mann-Whitney-Test relating to *LR_TEXT_ONLY*.

A.4 Conclusion

Both, LIWC as well as SentiWS are able to enhance the performance of a logistic regression classifier whereas a combination of both data sources yields the best results. In the setup, that is described in this section, SentiWS was used as a simple word list. The polarity weights of SentiWS were not used. Further research could examine in which configuration sentiment lexica can be used the most efficient way.

Eidesstattliche Erklärung
Ich versichere an Eides statt, dass ich die vorliegende Arbeit selbständig angefertigt und mich keiner fremden Hilfe bedient sowie keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen, die wörtlich oder sinngemäß veröffentlichten oder nicht veröffentlichten Schriften und anderen Quellen entnommen sind, habe ich als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Dortmund, den 29.8.2017

Leonard Hövelmann

Erklärung
Mir ist bekannt, dass nach § 156 StGB bzw. § 161 StGB eine falsche Versicherung an Eides Statt bzw. eine fahrlässige falsche Versicherung an Eides Statt mit Freiheitsstrafe bis zu drei Jahren bzw. bis zu einem Jahr oder mit Geldstrafe bestraft werden kann.

Dortmund, den 29.8.2017

Leonard Hövelmann