Learning from Few Samples: Lexical Substitution with Word Embeddings for Short Text Classification

Ábel Elekes
Karlsruhe Institute of Technology
Am Fasanengarten 5
Karlsruhe, Germany 76128
abel.elekes@kit.edu

Simone Di Stefano
Karlsruhe Institute of Technology / Echobot Gmbh
distefano@echobot.de

Martin Schäler
Karlsruhe Institute of Technology
Am Fasanengarten 5
Karlsruhe, Germany 76128
martin.schaeler@kit.edu

Klemens Böhm
Karlsruhe Institute of Technology
Am Fasanengarten 5
Karlsruhe, Germany 76128
klemens.boehm@kit.edu

Matthias Keller
Echobot Gmbh.
Karlsruhe, Germany
matthias.keller@echobot.de

ABSTRACT

Text classification helps to categorize large number of documents in Digital Libraries. Text classification results highly depend on the quality of labeled training data. In practice, the process of manually annotating documents is a hidden cost that is often overlooked. We propose a general preprocessing method for scenarios in which training data is scarce. It clusters semantically similar terms by including both a semantic distance measure and a probabilistic model of any task-specific term distributions. By preprocessing the training data with our method, one increases the mean classification performance of *all* tested classification approaches in text classification tasks having 500 or 1000 training samples. The largest observed increase is 15%. When more training samples are available, we report significant improvements in most scenarios as well.

KEYWORDS

Text Classification, Word Embedding models, Lexical Substitution

ACM Reference format:
Ábel Elekes, Simone Di Stefano, Martin Schäler, Klemens Böhm, and Matthias Keller. 2019. Learning from Few Samples: Lexical Substitution with Word Embeddings for Short Text Classification. In *Proceedings of ACM SIGIR, Paris, France, July 2019 (SIGIR 2019),* 8 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Text classification (TC), i.e., to assign predefined categories to text documents [14], is a highly relevant task in Digital Libraries, since it is an effective way to organize enormous number of documents. [29] [2] [28] It has other important applications, like spam filtering [15] or health prediction [21]. Various companies offer services to classify documents such as Twitter feeds for relevant signals, like risk factors or new trends.

Nowadays, TC is mainly done by machine-learning (ML) approaches [22]. One can train classifiers for an arbitrary class and expect good results as long as there is sufficient training (i.e., labeled) data [14]. However, the costs for generating training data for ML solutions are often significant and rarely discussed. E.g., the recent AI breakthroughs such as autonomous driving are based on enormous human tagging effort[5]. In the field of TC, academic research focuses on improving the classification accuracy of annotated benchmark data-sets for comparability reasons. But, in practice, the generation of such a data set and the necessary size are important cost factors. Hence, the costs of generating large bodies of labeled data for all potentially relevant classes are a severe issue in many TC use cases: The more data a classifier uses for learning, the better it tends to predict classes.

An alternative TC approach is the way how humans categorize text, by taking knowledge on semantic relationships between words into account. Word embedding models [16] – in theory – contain such relationships. So we wonder: How to extend TC approaches with exogenous knowledge on word semantics contained in such models to compensate the scarcity of labeled data? We target at a general *preprocessing* step compensating the scarcity of labeled data and ultimately increasing text classification accuracy.

Challenges. The complexity and variability of human language makes TC a non-trivial task, in particular if there is not much labeled data, which is usually the case in Digital Libraries [20]. Existing TC approaches may represent a document in a feature space as a multi-dimensional vector – with one dimension per existing word. Even for small datasets, this representation can have thousands of dimensions. This results in two issues, which reduce the accuracy. First, classifiers trained with small data samples lack robust statistical information and tend to *overfit* the training data. Second, unknown documents very likely contain words not covered by the classification model - a.k.a. *out of vocabulary (OOV)* words.

Next, the semantic relationships of word embedding models need to be considered with care. Namely, not only words with a similar meaning are semantically related, but also antonyms: Words like *'good'* and *'bad'*, which obviously are not interchangeable, tend to be very similar in word embedding models. Thus, approaches such as [13] that simply replace OOV words with the most similar labeled one, have little or even negative effect on classification accuracy.

Contributions. In this work, we present a lexical substitution approach for preprocessing that compensates the scarcity of labeled data. It is an orthogonal extension to virtually any existing TC approach, to improve classification accuracy. Our approach mimics how human annotators preprocess texts. It *replaces* words unknown to the classifier with known ones and *substitutes* semantically similar words for statistical robustness, based on the main contribution of this paper: a novel *semantic-distributional word distance measure*. Our preprocessing approach is a contribution to TC for the following reasons:

(1) It combines semantic and distributional information of words.
(2) It uses Bayesian Hypothesis Testing to decide when to substitute a word.
(3) It is generally applicable in combination with any TC algorithm, because we only replace words in the training data.
(4) It improves classification accuracy in all evaluated datasets when trained on at most 1000 samples and in most scenarios with bigger samples as well.

The novelty of our approach lies in the combination of semantic and distributional similarity. As a result, even if there is plenty of labeled data for learning, we observe better classification results than without preprocessing in various tasks. Nevertheless, the improvement in classification accuracy with our method is most significant when labeled data is scarce.

2 RELATED WORK

The core idea of our method is *term extraction* [22]. It clusters terms to reduce the dimensionality of the feature space while incorporating semantic information. This means that we create clusters of terms (dictionaries) which contain semantically similar words and then substitute words in a cluster with a word representing it. Previous methods either use distributional information (*Distributional Clustering*) or information on semantic relatedness (*Semantic Clustering*).

2.1 Distributional Clustering

Baker proposes to cluster terms using a distributional metric based on a variant of the Kullback-Leibler divergence [3]. The metric expresses the discriminative information loss when clustering two terms together. Their experiments have shown that feature dimensionality can be heavily reduced without losing much classification accuracy. Nonetheless, their evaluations have not shown any improvement over unprocessed datasets. Based on [3], Slonim and Tishby [23] propose an improved clustering algorithm using the *Information Bottleneck Method.* They maximize the mutual information of a word and its cluster with respect to their relative distribution over the categories. They report accuracy improvements of up to 18% when using the same number of cluster features instead of word features. However, such significant differences between the word- and cluster-based approaches are only observed with an extremely reduced set of features (as few as 25). [7] focuses on a reduced set of features as well, while [4] do report improvements for only one out of three tested datasets when applying distributional clustering. [1] achieve significant improvements over an SVM-based method using distributional clustering and a learning logic technique, but it is unclear whether these improvements are due to the distributional information or to other effects.

2.2 Semantic Clustering

With the popularity of word embeddings, semantic clustering has been widely researched. [13] uses the cosine similarity between word vectors to cluster terms using K-Means. Their results suggest that certain values of K might yield a slightly better classification. [26] use word-embedding-based clustering in combination with neural networks. They use the Euclidean distance in the embedding vector space to map similar phrases (n-grams) to the same neural units. Classification has been evaluated on short texts. Their work reports slight improvements in classification over other approaches.

Recently it has become common to use pre-trained word embeddings and to fine-tune them to integrate task-specific information [11] [6]. However, this does not work at all for smaller data sets. This is because fine-tuning changes only the embeddings seen in the training data. Hence it distorts the word structure of the pre-trained model when using small training data sets. This ultimately hurts its generalization performance [8] [31].

2.3 Implications

As opposed to our method, any existing term clustering approach refers to one group. In other words, they do not integrate language semantics or do not consider the distributional properties of the classification task. As our experiments in Section 4 show, including both kinds of information is very suitable for short text classification and is crucial for consistent improvements in TC accuracy. It also complements the novel fine-tuning approaches well: In contrast to them, our method is most effective with small training data.

Moreover, there is another advantage of combining the two clustering methods which is highly relevant in the Digital Library community, described in the following. Embedding models contain neural networks with millions of parameters. This makes understanding their clustering decisions just by evaluating the underlying model virtually impossible. In contrast, this is not the case for distributional methods where we can clearly interpret why two words are clustered together or not based on the distributional statistics. Such with the combination of the two approaches we are able to employ the robustness of embedding models while retaining the explainability of the method to the user.

3 METHOD

There are various real-world problems where labeled training data is scarce. An inherent limitation of any TC method that does not rely on external knowledge is that, when applied to unseen samples, it can only gain robust information from words that appear in the training data frequently enough. One option to increase classification accuracy is to increase the number of labeled documents. But this can be costly or even infeasible.

Cluster 1	+	-	Cluster 2	+	-	Cluster 3	+	-
anarchical	0	1	absurdly	0	1	*illegitimate*	0	12
capitalism	0	1	admittedly	1	0	*legitimate*	35	9
colonialism	0	4	amazingly	1	0	ruse	0	1
...			...			...		

Table 1: Clusters Generated with Naive Approach

Cluster 1	+	-	Cluster 2			Cluster 3	+	-
anarchical	0	1	genuine	1	0	illegality	0	1
imperialism	2	1	*legitimate*	35	9	illegitimacy	0	1
imperialist	0	4	logical	3	3	*illegitimate*	0	12
...			...			...		

Table 2: Clusters Generated with Our Method

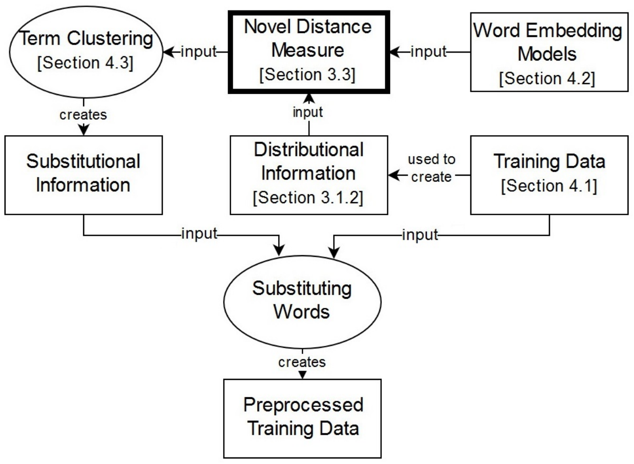
Term Clustering [Section 4.3]
input
Novel Distance Measure [Section 3.3]
input
Word Embedding Models [Section 4.2]
creates
input
Substitutional Information
Distributional Information [Section 3.1.2]
used to create
Training Data [Section 4.1]
input
input
Substituting Words
creates
Preprocessed Training Data

Figure 1: The Preprocessing Method

3.1 Intuition

The intuition behind our method is to improve the ability of TC algorithms to generalize. We do so by transforming the input data so that task-specific synonyms are merged. Task-specific synonyms are words that are used synonymously with respect to the classification task. All words in a group of synonyms are mapped onto one and the same representative word. So unseen words can be handled, and random effects due to words with few occurrences in the training data are reduced, as shown in the following example.

Example 1. Consider a sentence-classification task, namely that we want to detect changes in company management. A classifier might fail to classify “The corporation announced a new CEO for 2017” as positive, even if it has seen the sentence “The company announced a new executive for 2017” in the training data, because it is unaware of synonyms.

A straightforward approach is to use word embedding models and cluster words based on their semantic similarity, however this naive approach has severe limitations as explained in the next section.

3.1.1 Limitations of Naive Approaches. [13] report minor improvements when clustering words based on word similarity alone, and our own evaluations on different datasets were disappointing. TC accuracy often has been worse than with the original datasets. We see two reasons for this: The first one is the inability to distinguish antonyms and synonyms with the word-embedding model. Second, there is the problem of task-specific synonyms. While the first problem could be addressed using more sophisticated methods to generate word embeddings [25] or to detect antonyms [18], the second one is inherent to the outlined approach. Consider again Example 1 of detecting changes in management, a real-life classification problem that we studied together with an industry partner. In this specific case, we do not tag changes of project manager positions as positive in the training data, since our task is on identifying fluctuations in *upper* management. Hence, even if they are close in the embedding space, we would not want to merge the word ‘manager’ with other management positions that are relevant, such as ‘CEO’. In this case ‘manager’ and ‘CEO’ are not considered synonyms. Note that this is only due to this specific classification task. In other scenarios in turn, it would make sense to merge these job names, i.e., consider them as task-specific synonyms.

3.1.2 Integrating Distributional Information. We found that we can address both problems described above, i.e., antonyms and task-specific synonyms, by integrating information on how the words are distributed among the positive and negative training examples. Consider the clusters that the naïve method has generated on the MPQA dataset shown in Table 1. ’illegitimate’ and ’legitimate’ are part of the same cluster, even if they clearly are no task-specific synonyms for distinguishing positive and negative expressions. The table also shows how often each word appears in a negative and positive training sample. It is obvious that illegitimate and legitimate are significantly differently distributed among the positive and negative samples – a strong indication that they are not used synonymously in this task. In the following we describe a probabilistic way of weighting the distributional indications against a probably imperfect measure for semantic relatedness. Table 2 shows the clusters our method generates. Note that the words imperialist and imperialism are not merged, even if imperialist appears more often in the positive class, while imperialism has more occurrences in the negative class. This illustrates the strength of our probabilistic approach: Due to the few counts for both words, the difference in the distribution is not significant enough and does not outweigh the low semantic distance in our formula.

3.2 Processing Overview

Figure 1 shows the proposed preprocessing method. The notation follows the Yourdon/DeMarco notation standard. Boxes represent input/output data, ellipses functions and stripes databases. Arrows represent the flow between the states or data. The sections where we describe the steps/the data are in brackets.

Our novel distance measure (derived in Section 3.3) is defined for each word pair that appears in the training data and the pretrained word-embedding space. A small value stands for a high probability of the two words being task-specific synonyms. To replace synonyms with a (random) representative word, a clustering algorithm is applied to the distance matrix over the vocabulary. OOV-words are replaced by the representative word of the nearest cluster if the distance is under a predefined threshold value.

3.3 Deriving a Semantic-Distributional Word Distance Measure

The derivation of our measure consists of the following steps.

(1) Modeling of word occurrences as being generated by a Bernoulli process. (Section 3.3.1)
(2) Interpretation of assigning words to the same cluster as a probabilistic hypothesis. (Section 3.3.2)
(3) Using *Bayesian Hypothesis Testing* to assess the plausibility of (2). (Section 3.3.3)
(4) Incorporating semantic information by using Bayesian priors. (Section 3.3.4)

3.3.1 Word Occurrences as a Bernoulli Process. To derive the distributional dissimilarity between pairs of words, it is common to view word occurrences as being generated by a probabilistic process. A probabilistic model commonly used to describe word occurrences distributed over a dichotomy of classes is the *Bernoulli process* [12]. Suppose that a word $v \in V$ appears n times in a labeled dataset D. X is a random variable representing the occurrences of v in the positive class. Assuming an underlying Bernoulli process, the probability that, for n total occurrences, v appears k times in the positive class is:

$$B(n,k,\theta) := \Pr(X = k;\ \theta) = \binom{n}{k}\theta^k(1-\theta)^{n-k}$$

where $\theta \in [0,1]$ is the distribution parameter. For a random variable drawn from a Bernoulli process with n trials, we also use the notation

$$X \sim B(n,\theta)$$

3.3.2 Merging Words as a Probabilistic Event. We now perceive the decision when to assign two words to the same cluster as a probabilistic event. Ultimately, this implies modeling the decision of how useful it is to the underlying classification task to substitute a word v_1 with a word v_2 or vice versa. From now on, instead of speaking of “substituting words”, we speak of “merging words”, since the substitution direction is irrelevant.

We now model the decision process (*merging* or *not merging*) probabilistically. To do so, we associate either decision with the probabilities of two hypotheses H_0, H_1. They represent the plausibility of merging or not merging respectively.

Hypothesis Formulation. Let X_i, X_j be the random variables that represent the occurrences of $v_i, v_j \in V$. Let n_i, n_j be the total number of occurrences observed for v_i, v_j respectively.

Hypothesis H_0. (No Merge) In H_0, we assume that the observed word occurrences X_i, X_j are generated by two distinct latent, independent Bernoulli processes, i.e.,

$$H_0 : X_i \sim B(n_i,\theta_1), X_j \sim B(n_j,\theta_2), \theta_1 \neq \theta_2$$

In this scenario, we do not assume any relatedness of X_i, X_j, i.e., we do not merge v_1 and v_2.

Hypothesis H_1. (Merge) The probability of the second hypothesis H_1 should express how well we can explain the observed data if we assume that a common, latent generative process generates the occurrences of both words X_i, X_j:

$$H_1 : X_i \sim B(n_i,\theta_1), X_j \sim B(n_j,\theta_2), \theta_1 = \theta_2$$

High probabilities indicate that v_i and v_j are generated by similar processes and can be merged without losing any discriminatory information.

The goal now is to calculate the probabilities of the competing hypotheses H_0, H_1, given the observed data. We do this in Section 3.3.3 using *Bayesian Hypothesis Testing*. But before applying such methods, we establish estimates for the parameters (θ) of the underlying Bernoulli processes.

Maximum Likelihood Estimation. When there is no further knowledge about the generative process, a commonly used method to estimate the parameters of the underlying distribution is Maximum Likelihood Estimation (MLE). As shown in [19], in case of a Bernoulli process with n trials and k successes, the MLE is

$$\hat{\theta}_{MLE} = \arg\max_{\theta'} B(n,k,\theta') = \frac{k}{n}$$

Let k_i, k_j be the numbers of occurrences of v_i, v_j in the positive class. For H_0, the MLEs for each Bernoulli process generating X_i are the following ones:

$$H_0 : \hat{\theta}_{H_0,i} = \frac{k_i}{n_i}$$

For the second hypothesis, we assume that the independent observations X_i, X_j have the same underlying distribution parameter. Assuming that this hypothesis is true, the MLE for H_1 is

$$H_1 : \hat{\theta}_{H_1} = \hat{\theta}_{H_1,1} = \hat{\theta}_{H_1,2} = \frac{k_i + k_j}{n_i + n_j}$$

In the next section we describe how to use *Bayesian Hypothesis Testing* to assess the probability of either hypothesis.

3.3.3 Bayesian Hypothesis Testing. Bayesian Hypothesis Testing is a tool to estimate the probability of competing models. In our context, we want to compare how well the occurrences of two words v_i, v_j in a document set D is explained before and after merging these, with decisions being represented by H_0 and H_1. We start with the proposition that either H_0 or H_1 are true, i.e.,

$$\Pr(H_0 \vee H_1 \mid D) = \Pr(H_0 \mid D) + \Pr(H_1 \mid D) = 1 \tag{1}$$

The probability of H_0, i.e., two different Bernoulli processes generate the words, can be expressed using the Bayesian Rule:

$$\Pr(H_0 \mid D) = \frac{\Pr(D \mid H_0)\Pr(H_0)}{\Pr(H_0)\Pr(D \mid H_0) + \Pr(H_1)\Pr(D \mid H_1)} \tag{2}$$

$$\Pr(H_1 \mid D) \overset{(1)}{=} 1 - \Pr(H_0 \mid D)$$

Note that, since Hypotheses H_0 and H_1 are complementary, computing the probability of either one is sufficient. Without loss of generality, we now only use $\Pr(H_0 \mid D)$ since it also is a dissimilarity measure, i.e., higher probabilities mean higher distributional incompatibility. We first simplify $\Pr(H_0 \mid D)$ as follows:

$$\begin{aligned}\Pr(H_0 \mid D) &= \frac{\Pr(D \mid H_0)\Pr(H_0)}{\Pr(H_0)\Pr(D \mid H_0) + \Pr(H_1)\Pr(D \mid H_1)} \\ &= \frac{1}{1 + \underbrace{\frac{\Pr(H_1)}{\Pr(H_0)}\frac{\Pr(D \mid H_1)}{\Pr(D \mid H_0)}}_{=:\kappa}} = (1+\kappa)^{-1}\end{aligned}$$

First we derive an estimate for κ and then we will insert it back in the estimate for $\Pr(H_0 \mid D)$ at the end of our argumentation. κ is called the *Bayes Factor* and is often used as an alternative to the probability to express the plausibility of H_0 over H_1. From now on, we will refer to $\Pr(H_0 \mid D)$ as *semantic-distributional dissimilarity* or *semantic-distributional distance.*

3.3.4 Estimation of the Semantic-Distributional Distance Measure.

In this section, we derive an estimate for each probability of the Bayesian model, i.e., each probability on the right side of Equations 2. We will denote probability estimates with ˆ as follows:

$$\hat{P}(H_0 \mid D) \coloneqq (1+\hat{\kappa})^{-1}\,; \qquad \hat{\kappa} \coloneqq \frac{\hat{P}(H_1)\hat{P}(D \mid H_1)}{\hat{P}(H_0)\hat{P}(D \mid H_0)}$$

Moreover, k_i, n_i, k_j, n_j denote the word frequencies in the positive class and in the whole dataset of v_i and v_j respectively.

Estimation of model likelihoods. We proceed by using *Bayesian Hypothesis Testing* to derive estimates for the probability $\Pr(H_0 \mid D)$. To do so, we estimate the likelihoods $\Pr(D \mid H_i)$, $i = 0, 1$. We make the following assumptions:

(1) Word occurrences follow a Bernoulli distribution with the probability-density function $B(n, k, \theta)$
(2) For word co-occurrences we assume conditional independence (c. i.).

These are common, realistic assumption in NLP research [10] [9]. We parametrize the Bernoulli distribution with the Maximum likelihood estimates from before. For $\Pr(D \mid H_0)$, the estimate is:[1]

$$\begin{aligned}&\hat{P}\left(X_i = k_i, X_j = k_j \mid \theta_1 = \hat{\theta}_{H_0,1}, \theta_2 = \hat{\theta}_{H_0,2}\right) \\ &\overset{\text{c.i.}}{=} B(n_i, k_i, \theta_{H_0,1}) \cdot B(n_j, k_j, \theta_{H_0,2}) \\ &= \binom{n_i}{k_i}\hat{\theta}_{H_0,1}^{k_i}\left(1-\hat{\theta}_{H_0,1}\right)^{n_i-k_i} \cdot \\ &\quad \binom{n_j}{k_j}\hat{\theta}_{H_0,2}^{k_j}\left(1-\hat{\theta}_{H_0,2}\right)^{n_j-k_j}\end{aligned}$$

Analogously, for $\Pr(D \mid H_1)$ the estimate is

$$\hat{P}(D \mid H_1) = \binom{n_i}{k_i}\binom{n_j}{k_j}\hat{\theta}_{H_1}^{k_i+k_j}\left(1-\hat{\theta}_{H_1}\right)^{n_i+n_j-k_i-k_j}$$

We then insert $\hat{P}(D \mid H_i)$, $i = 0, 1$, in the estimate $\hat{\kappa}$ for κ

$$\hat{\kappa} = \frac{\hat{P}(H_1)}{\hat{P}(H_0)} \cdot \frac{\hat{\theta}_{H_1}^{k_i+k_j}\left(1-\hat{\theta}_{H_1}\right)^{n_i+n_j-k_i-k_j}}{\hat{\theta}_{H_0,1}^{k_i}\left(1-\hat{\theta}_{H_0,1}\right)^{n_i-k_i}\hat{\theta}_{H_0,2}^{k_j}\left(1-\hat{\theta}_{H_0,2}\right)^{n_j-k_j}}$$

[1]We consider only the probabilities for v_i and v_j, since all other word probabilities are equal for both hypotheses and do not affect the calculation of $\hat{\kappa}$

Integration of semantic knowledge in the priors. The only probabilities left for estimation are the priors $\Pr(H_0)$, $\Pr(H_1)$. In Bayesian models, the prior probabilities are often used as an interface to incorporate prior, expert knowledge in the models. When there is no further knowledge about the prior probabilities $\Pr(H_0)$ and $\Pr(H_1)$, one commonly assumes that every hypothesis is equally probable, i.e., $\hat{P}(H_0) = \hat{P}(H_1) = 0.5$ [30]. But we can use the information provided by word-embedding models to approximate the priors. The calculation of word vectors implies predicting the probability of a word appearing in a given context. The cosine similarity of two word vectors can be interpreted as an approximation of the probability that two words v_i, v_j appear in similar contexts [16, 17]. It ranges from -1 (unrelated words) to 1 (identical words). If the cosine similarity expresses the relatedness between two words, the *cosine distance* values can be used for dissimilarity. For an embedding $\mathcal{W}$ we define the following distance measure:

$$dist_{\mathcal{W}}(v, v') = \frac{1 - \text{cos-sim}(vec_{\mathcal{W}}(v), vec_{\mathcal{W}}(v'))}{2} \in [0, 1],$$

where $vec_{\mathcal{W}}(v)$ represents the vector corresponding to v in embedding $\mathcal{W}$.

For words farther away in the embedding vector space we favor the first hypothesis, i.e., the words should not be merged. We therefore introduce a parameter $\alpha \in [0, 1]$ that specifies how much the cosine distance lets the priors deviate from 0.5.

$$\begin{aligned}\hat{P}(H_0; \alpha) &\coloneqq \alpha \cdot \left(\frac{1}{2} - dist_{\mathcal{W}}(v_i, v_j)\right) + \frac{1}{2} \\ \hat{P}(H_1; \alpha) &\coloneqq 1 - \hat{P}(H_0; \alpha)\end{aligned}$$

With parameter α included in the equations, we can tune our preprocessing procedure. For $\alpha = 0$, the cosine distance is completely neglected, and the hypotheses are assumed to be equally probable. For $\alpha > 0$, the priors deviate from 0.5 proportionally to $\alpha \cdot dist(v_i, v_j)$. In Section 4 we calculate the classification accuracy with different values of α to find the best one for each dataset.

After inserting the above priors in $\hat{\kappa}$, we obtain:

$$\hat{\kappa} = \kappa(\alpha) = \frac{\frac{1}{2} - \alpha \cdot \left(\frac{1}{2} - dist_{\mathcal{W}}(v_i, v_j)\right)}{\alpha \cdot \left(\frac{1}{2} - dist_{\mathcal{W}}(v_i, v_j)\right) + \frac{1}{2}} \cdot \frac{\hat{P}(D \mid H_1)}{\hat{P}(D \mid H_0)}$$

Finally, we define the *semantic-distributional distance* that incorporates both distributional and semantic information on the words:

$$\Lambda_{\alpha,\mathcal{W}}(v_i, v_j) \coloneqq \hat{P}(H_0 \mid D) = (1 + \hat{\kappa}(\alpha))^{-1}$$

We note that $\Lambda_{\alpha,\mathcal{W}}(v_i, v_j)$ is defined using the cosine distance of word vectors in an embedding $\mathcal{W}$. Therefore, in order to produce valid distances, v_i and v_j must be part of the vocabulary $voc(\mathcal{W})$.

3.3.5 Correction Term.

Although the cosine similarity of two word vectors can be interpreted as an approximation of the probability of both words appearing in similar contexts, the method could be improved by learning a function that maps the cosine distance onto a more accurate probability estimation for words being synonyms. However, since we found the cosine similarity part to be too pessimistic, we add a correction term allowing for more weight

Dataset	$\|D\|$	$\|V\|$	ℓ	$\|D_+\|$	$\|D_-\|$
MPQA	10,624	6,298	3	3,316	7,308
CR	3,772	6,596	20	2,406	1,366
MR	10,662	20,621	21	5,331	5,331
Subj.	10,000	23,187	24	5,000	5,000

Table 3: Dataset statistics. Legend: $|D|$: size of dataset, $|V|$: number of words, ℓ: average document length, $|D_+|$: positive samples, $|D_-|$: negative samples.

in the final measure. An additional parameter $\beta \in [0, 1]$ controls the influence of this term:

$$\Lambda_{\alpha,\beta,\mathcal{W}}(v_i, v_j) = \beta\Lambda_{\alpha,\mathcal{W}}(v_i, v_j) + (1 - \beta)dist_{\mathcal{W}}(v_i, v_j)$$

4 EVALUATION

In this section we evaluate our method on different classification tasks. Accuracy is measured for different classifiers trained on unprocessed and preprocessed training data. We show that classifiers trained on our preprocessed data consistently outperform the ones trained on the original datasets.

4.1 Evaluation Datasets

To evaluate the effectiveness of the proposed method, we conduct experiments on short-text datasets benchmarked in numerous NLP studies: Customer Reviews (CR), MPQA, Short Movie Reviews (Rt10k) and Subjectivity (Subj). [2] A summary of additional information about the employed datasets can be seen in Table 3. All these datasets are cases of *binary classification*.

4.2 Classifiers

We use three classifiers for our evaluations. Our goal is to show that our preprocessing method increases the accuracy with various classifiers built on top of the preprocessed training data. To show this, we deploy the same classifiers used in previous studies [27]. We only change the respective training data to our preprocessed version. We use two baseline TC methods, the Multinomial Naive Bayes (MNB) and the Naive Bayes Support Vector Machine (NBSVM) classifiers. We parameterize them as recommended in [27].

The third classifier we evaluate is presented in [24]. It is based on recursive auto-encoders (RAEs) and works on the sentence level. At the time it was introduced it has produced state-of-the-art classification accuracy on various datasets. We initialize the word embeddings of the RAE randomly, as suggested by the authors.[3]

4.3 Pretrained Word Embeddings

We use Google's pretrained *word2vec* SG model for the evaluation. It has been trained on a part of Google's News dataset, which contains around 100 billion words. The final model consists of $3,000,000$ word vectors of dimensionality 300.[4]

[2]We have downloaded the datasets from https://github.com/sidaw/nbsvm.
[3]A MATLAB implementation of the classifier is available at http://www.socher.org.
[4]Details and download at https://code.google.com/archive/p/word2vec/

4.4 Term Clustering

The term clusters are computed with the built-in agglomerative hierarchical clustering algorithm in Matlab. [5] Its advantage over, say, K-Means is that it allows to operate on a dissimilarity matrix based on any distance function designed by the user. In our case, this is the semantic-distributional distance. However, any other algorithm that supports custom distance metrics could have been used (e.g., DBSCAN).

4.5 Experimental Setup

4.5.1 Methodology. We subdivide each dataset as follows: The *test sets* consist of 1000 samples held out from each dataset for later testing. To show the effectiveness of our method on different training-set sizes, parameter tuning and classifier training is performed with training sets of varying sizes: 500, 1000, 1500, 8500 for *MPQA*, *Rt10k* and *Subj* and 500, 1000, 1500, 2600 for *CR*. For each size, we sample five training sets randomly, using stratified sampling. For each of these sets, a 10-fold cross-validation is performed to find the best parameter combination, i.e., the combination that yields the highest average accuracy over all folds. The classifier is then trained on the same dataset that is used for the cross-validation and tested on the held-out test set (five times for each sample size and classifier combination). We report on these results in Section 4.6.

4.5.2 Parameter Search. During parameter tuning, we search on all combinations of the following parameter values,

$$\begin{aligned}\boldsymbol{K} &= \{0, 0.25, 0.5, 0.75, 0.9\}\\ \boldsymbol{\alpha} &= \{0, 0.25, 0.5, 0.75, 1\}\\ \boldsymbol{\beta} &= \{0, 0.1, 0.3, 0.5, 0.7, 1\}\\ \boldsymbol{\theta} &= \{0, 0.5, 2\}\end{aligned}$$

We exclude all combinations with $\beta = 0$ and $\alpha \neq 0$ (see Section 3.3.5). Instead of the absolute number of clusters, we use K as a fraction of terms the resulting preprocessed dataset is reduced to. E.g., $K = 0.25$ means that the unigrams are reduced to $0.25 \cdot |V|$, where $|V|$ is the number of unigrams in the dataset.

4.6 Results and Discussion

Figure 2 shows our evaluation results with three different classifiers. The horizontal axis represents the size of the training set. The vertical axis represents the difference between classification accuracy *with* lexical substitution and *without*, i.e., a positive value indicates improvement of our method over unprocessed datasets. [6] The red values (left bar) represent the accuracy difference using only semantic clustering, i.e., not using the distributional information of the training data. The blue values (right bar) correspond to the runs using our novel distance measure. The dots indicate the mean accuracy difference over five runs, the thick error bars stand for the corresponding 25 percentile mark, i.e., the accuracy difference of 3 of 5 runs. The thin lines extend to the minimum/maximum.

We report on the results achieved with the best parameter combination, as described previously. The accuracies used to calculate the difference in performance are the mean accuracies measured over five runs of our algorithm. The algorithm is run with 4 different training sizes. We can observe average improvements in all

[5]https://www.mathworks.com/help/stats/cluster.html
[6]Note, that the value ranges are different between the rows.

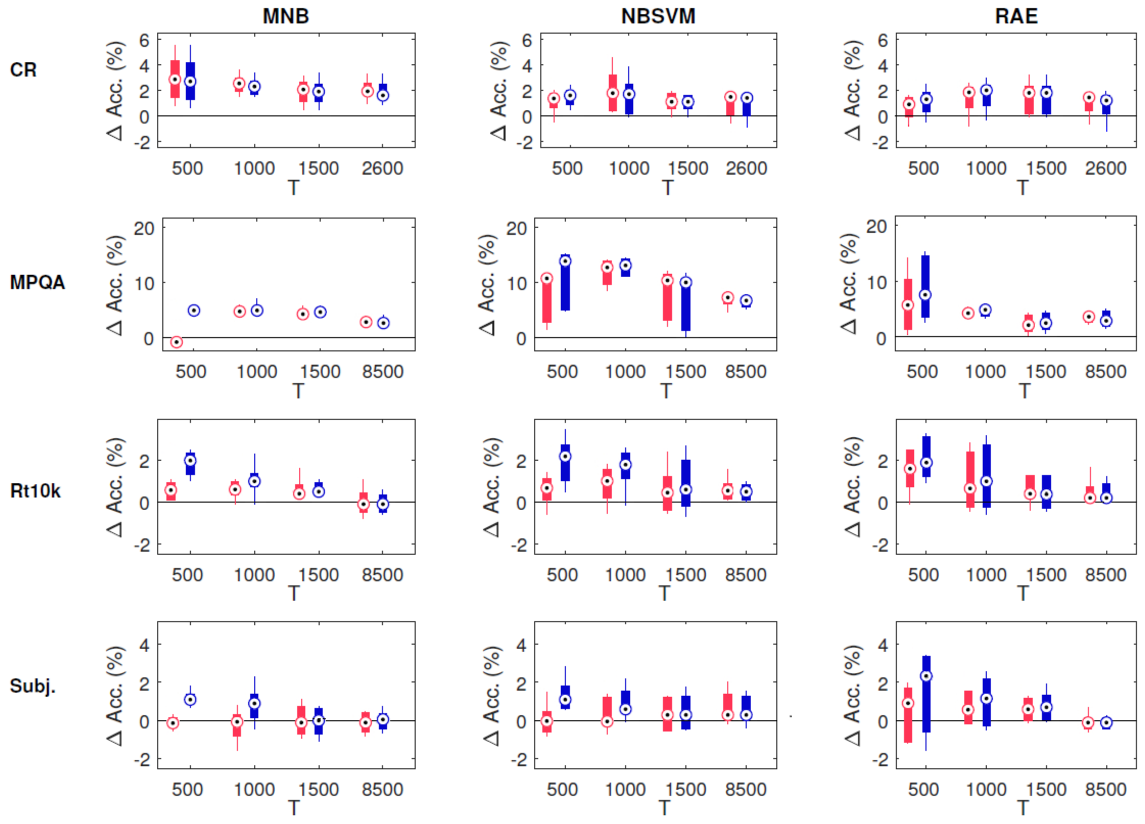

Figure 2: Classification accuracy for each dataset (rows) using three different classifiers (columns).

datasets, except for the *Subj* dataset classified with RAE and the *Rt10k* dataset classified with MNB, both trained with 8500 samples. More precisely, of all 240 scenarios tested[7], in 197 cases (82 %) our method improves classification accuracy, in 5 cases (2 %) no change is observed, and in 38 cases (16 %) the performance slightly deteriorates. In 173 cases (72 %) distributional information improves classification accuracy, in 11 cases (5 %) there is no change, and in 56 cases (23 %) the accuracy is worse. In 46 cases (19 %) distributional information has a higher added value in terms of classification accuracy than semantic information. These cases are mostly ones on the Subjectivity dataset and on smaller training sets.

As expected, the largest improvements are observed in the smaller training sets. The mean classification accuracy increases for *all* tested dataset and classifier combinations when the sample size is at most 1000 samples. This confirms the hypothesis that an exogenous knowledge base can improve classification when there is a lack of training samples. Next, even with more complex classifiers such as the RAE [24], our method could be used to facilitate the training of models which generalize and, hence, perform better.

We also observe higher improvements with growing training sets, e.g., for CR classified with RAE and NBSVM the biggest average quality jump is observed in training sets with 1000 samples. There seems to be an optimal training-set size, so that the external knowledge brought from word embeddings has the most beneficial effects. This is because distributional information becomes robust enough for the substitutions to be most efficient, compared to classification without word clusters.

[7] 4 Datasets $\times$ 4 Training-Set sizes $\times$ 5 Runs $\times$ 3 Classifiers = 240

5 CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a new approach to short text classification using lexical substitution based on word embedding models. The main contribution is the definition of a semantic-distributional word-distance measure. Together with a standard clustering algorithm, the measure can be used to preprocess input data for any existing TC algorithm. In our evaluation, the preprocessing increases the mean classification accuracy for any tested classifier and dataset combination when trained on 500 or 1000 samples. As an outlook, we expect further improvements by modelling the relationship between the word distance and the true probability of being synonyms more accurately.

REFERENCES

[1] Hisham Al-Mubaid and Syed A Umair. 2006. A new text categorization technique using distributional clustering and learning logic. *IEEE TKDE* 18 (2006).
[2] Henri Avancini, Andreas Rauber, Fabrizio Sebastiani, and TU Wien. 2004. *Organizing digital libraries by automated text categorization.* na.
[3] L Douglas Baker and Andrew Kachites McCallum. 1998. Distributional clustering of words for text classification. In *ACM SIGIR*. ACM.
[4] Ron Bekkerman et al. 2003. Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research* 3 (2003).

[5] Tim Bradshaw. 2017. Self-driving cars prove to be labour-intensive for humans. https://www.ft.com/content/36933cfc-620c-11e7-91a7-502f7ee26895. (07 2017). [Online; accessed 14-Jan-2019].
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[7] Inderjit S Dhillon, Subramanyam Mallela, and Rahul Kumar. 2002. Enhanced word clustering for hierarchical text classification. In *ACM SIGKDD*. ACM.
[8] Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies* 10, 1 (2017), 1–309.
[9] Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *UAI*.
[10] Thomas Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine learning* 42, 1 (2001), 177–196.
[11] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 328–339.
[12] Thorsten Joachims. 1998. Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. *ECML* (1998).
[13] Long Ma and Yanqing Zhang. 2015. Using Word2Vec to process big text data. In *Big Data (Big Data), 2015 IEEE International Conference on*.
[14] Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI*.
[15] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. 2006. Spam filtering with naive bayes-which naive bayes?. In *CEAS*, Vol. 17. 28–69.
[16] Tomas Mikolov et al. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.
[17] Tomas Mikolov et al. 2013. Efficient estimation of word representations in vector space. *arXiv:1301.3781* (2013).
[18] Nikola Mrkšić et al. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv:1603.00892* (2016).
[19] In Jae Myung. 2003. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology* 47, 1 (2003), 90–100.
[20] Kamal Nigam et al. 2000. Text classification from labeled and unlabeled documents using EM. *Machine learning* (2000).
[21] Sandip Ray et al. 2007. Classification and prediction of clinical Alzheimer's diagnosis based on plasma signaling proteins. *Nature medicine* 13 (2007).
[22] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)* 34, 1 (2002), 1–47.
[23] Noam Slonim and Naftali Tishby. 2001. The power of word clusters for text classification. In *ECIR*.
[24] Richard Socher et al. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*.
[25] Bruna Thalenberg. [n. d.]. Distinguishing Antonyms from Synonyms in Vector Space Models of Semantics. ([n. d.]).
[26] Peng Wang et al. 2016. Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing* 174 (2016).
[27] Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*.
[28] Tao Wang and Bipin C Desai. 2007. An approach for text categorization in digital library. In *Database Engineering and Applications Symposium, 2007. IDEAS 2007. 11th International*. IEEE, 21–27.
[29] Ian H Witten, Katherine J Don, Michael Dewsnip, and Valentin Tablan. 2004. Text mining in a digital library. *International Journal on Digital Libraries* 4, 1 (2004), 56–59.
[30] Shelemyahu Zacks. 1971. *The theory of statistical inference*. Vol. 34.
[31] Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820* (2015).