Lex2Sent: A bagging approach to unsupervised sentiment analysis

Kai-Robin Lange and Jonas Rieger and Carsten Jentsch
Department of Statistics, TU Dortmund University, 44221 Dortmund, Germany
{kalange, rieger, jentsch} @statistik.tu-dortmund.de

Abstract

Unsupervised text classification, with its most common form being sentiment analysis, used to be performed by counting words in a text that were stored in a lexicon, which assigns each word to one class or as a neutral word. In recent years, these lexicon-based methods fell out of favor and were replaced by computationally demanding fine-tuning techniques for encoder-only models such as BERT and zero-shot classification using decoder-only models such as GPT-4. In this paper, we propose an alternative approach: Lex2Sent, which provides improvement over classic lexicon methods but does not require any GPU or external hardware. To classify texts, we train embedding models to determine the distances between document embeddings and the embeddings of the parts of a suitable lexicon. We employ resampling, which results in a bagging effect, boosting the performance of the classification. We show that our model outperforms lexica and provides a basis for a high performing few-shot fine-tuning approach in the task of binary sentiment analysis.

1 Introduction

Most commonly, text classification is performed in a supervised manner by using a previously labeled data set to train a learning-based model to predict the sentiment of unlabeled documents. When a labeled data set is not available, an unsupervised labeling approach is useful to provide valuable initial information for an active learning approach or to label the texts right away, when a near-perfect classification is not strictly necessary. However, such unsupervised models often require financial backing or a high performing GPU to use on a large data set.

In this paper, we propose Lex2Sent, a model mainly designed for sentiment analysis, that can however be used for any binary text classification problem, where external resources in the form of lexica are available. We will thus define the model for any arbitrary binary classification. Lex2Sent uses text embedding models to estimate the similarity between a document and both halves of a given binary lexicon. These distances are calculated for multiple resampled corpora and are aggregated to achieve a bagging-effect. As Doc2Vec models are usually trained on the CPU, the method demonstrated here can be fully realized in low hardware resource environments that do not have access to a GPU or the financial means to let commercial models such as GPT label thousands of documents. As the Lex2Sent's architecture is not dependent on the language of choice, it can also be used in other languages than English, including low resource languages for which no powerful language models are available. To demonstrate that the results are generalizable, we compare them to the ones of traditional lexicon methods on three data sets with distinct characteristics. To assess the performance to the modern unsupervised classification state of the art, we compare Lex2Sent's results to GPT-3.5 on one data set. We also extend this active learning approach by fine-tuning a RoBERTa model on a sufficient subset of the labels predicted by Lex2Sent. This can be seen as an initial starting point for active learning approach.

The paper is structured as follows. In Section 2, we discuss previous approaches to text classification and research on resampling techniques for texts. Section 3 introduces our classification model by describing the Doc2Vec model, the unsupervised labeling approach and the resampling procedure used. The data sets and lexica used are specified in Section 4. In Section 5, the classification rates of Lex2Sent are compared to lexicon methods and the performance of Chat-GPT. We also show that we can use the results of Lex2Sent for an initial fine-tuning of a pre-trained language model in few-shot setting. In Section 6, we conclude and give an outlook to further research.

2 Related Work

When little to no labeled data is available, usually text classification is performed in one out of three ways. That is, by using either traditional lexicon methods, decoder-only models like GPT or parameter efficient fine-tuning methods to fine-tune pre-trained language models.

Traditionally, researchers used lexica/dictionaries that were meant to substitute the missing supervised label information by external information. For sentiment classification, such lexica contain both a list of positive and negative words, which could simply be counted within a text. Commonly used lexica include VADER (Hutto and Gilbert, 2014), Afinn (Nielsen, 2011), Loughran-McDonald (Loughran and McDonald, 2010), the WKWSCI lexicon (Khoo and Johnkhan, 2018) and the Opinion lexicon (Hu and Liu, 2004). Even within a specific task such as binary sentiment classification, these lexica are often designed for a specific use case. For instance, the Loughran-McDonald lexicon is designed for economic text data, while VADER is designed for social media data. Lange and Jentsch (2023) perform a sentiment analysis of German political speeches and use Lex2Sent with a lexicon base specifically designed for German political text data (Rauh, 2018).
This method is very resource-savvy, but yield worse performance than the other two methods. Nowadays, lexica are usually only used in low hardware resource environments or by researchers of social science disciplines, because they are white-box algorithms that are easy to interpret.

Alternatively, GPT-4 (Brown et al., 2020) or any other large language model (e.g. Llama 2 (Touvron et al., 2023), Mixtral (Jiang et al., 2024) or Jamba (Lieber et al., 2024)) can classify any document in a zero-shot manner due to their language understanding capabilities. Using GPT-4 or GPT-3.5 for large corpora requires financial backing not everyone has access to though and similarly, open source large language models need a GPU with large vram.

Lastly, a pre-trained Transformer model like BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019), that was additionally fine-tuned on the task at hand, might help when a GPU is available, that cannot handle a large language model. This however yields the downside of using classification rules that are not based on the texts the model is meant to be used on. Instead, the model might carry a bias from a different subject over to the classification: the sentiment of a text might be based on completely different clues based on whether the text is a political speech or a social media post. This can be avoided by fine-tuning the model oneself, which, in turn, needs labeled data. To reduce the amount of data needed, active learning (Tharwat and Schenck, 2023) is increasingly being used in combination with few-shot learning techniques. Parameter-efficient fine-tuning (PEFT, Mangrulkar et al., 2022) uses adapter methods such as Low Rank Adaption (LoRA, Hu et al., 2021) to fine-tune language models with fewer training parameters than usual, and is thus suited to fine-tune on few-shot examples to achieve adequate results. Pattern exploiting training (PET, Schick and Schütze, 2021) uses the language understanding capabilities of language models to its advantage by "explaining" the task to the model. As Rieger et al. (2024) show, such methods can even be effectively combined into one.

Contrary to these approaches, we propose a fully unsupervised approach that can be used in low hardware resource environments, in which no access to a GPU is available and where there is no financial backing to let commercial models like GPT label thousands of documents. We do this by employing CPU-based embedding algorithms that leverage external information using lexica and are further improved by resampling, resulting in a bagging effect. Improving embedding-based text classification with the help of lexica has been explored by Shin et al. (2017), Mathew et al. (2020a) and Mathew et al. (2020b), but neither analyze a combination of embeddings and lexica for unsupervised analysis.

Xie et al. (2020) use resampling to improve the performance of supervised sentiment models by resampling words with certain probabilities based on their tf-idf-score or by translating the original document into another language and then translating it back to the original language. Similar augmentations can be performed with the nlpaug-package (Ma, 2019). This allows the user to, for instance, use embedding models, be it static models like Word2Vec (Mikolov et al., 2013) or contextual masked language models like BERT (Devlin et al., 2019). These types of data augmentation and resampling are most often used as additional training data for the embedding models and supervised methods. In this paper, instead of resizing the training set, we create multiple different training sets, on which one embedding model is trained each. Aggregating the information from these models into one combined classifier creates a bagging effect, improving the classification rate. Furthermore, we investigate the advantage of using such augmentation and resampling techniques in an unsupervised setting.

The procedures used by Xie et al. (2020) and Ma (2019) do however change the existing vocabulary. They either change the vocabulary by back-translation or resampling the document dependently from other documents due to the tf-idf-scoring or even introduce completely new words that are not part of the corpus at all by changing words based on similiar words in a given embedding space. This might be counter-productive for an unsupervised analysis, as the texts are not used as a training data set, but are supposed to be evaluated themselves. Changing the vocabulary might introduce a bias and hinder the classification performance, as the external information provided to classify the texts is given by the lexica, which are essentially word lists and thus more likely to work accurately to work with unchanged vocabulary. The resampling procedures in this paper are instead based on those employed by Rieger et al. (2020), who used resampling procedures to analyze the statistical uncertainty of the topic modeling method Latent Dirichlet Allocation. We chose those procedures, as they augment or resample the texts independently from another and do not add new words to the vocabulary.

3 Lex2Sent

In this section we propose Lex2Sent, a bagging model for unsupervised sentiment analysis. Lex2Sent is published as a Python package. The code can be found on GitHub[1].

3.1 Lexica

To perform unsupervised text classification, lexica can be used to interpret the words in a text without the need for previously labeled documents of a similar corpus, as they provide external information. This information is provided in the form of key words, which a lexicon assigns to a certain class.

For our analysis, we use binary lexica, that are used to separate words between two disjoint classes A and B. Such a lexicon assigns a value from an interval $[-s, s]$ with $s \in \mathbb{R}^+$ to all words, while assigning the value 0 to all neutral words. It assigns positive values to all words it deems to belong to class A and negative values to all words, it deems to belong to class B. To enable some words to have a larger weight during the classification process, lexica might give words different values. For instance, the word "fantastic" might receive a higher score than the word "good" when using a sentiment analysis lexicon, as it conveys an even stronger positive emotion. We modify such a binary lexicon to consist of two halves, one for each of the two classes. These halves are defined as lists of words in a way that each word that belongs to either class A or class B occurs exactly once in its respective half. Only neutral words are not assigned to a half. This enables the use of lexicon-based text embeddings.

As we use static embeddings, a key word's embedding is not changed, even if it is negated in a document. To incorporate the concept of negations into Lex2Sent, we merge negations with the following word during preprocessing. The term "not bad" is thus changed to "negbad". "negbad" is then added to the opposite lexicon half of the word "bad", so that Lex2Sent can interpret it correctly.

3.2 Lexicon-based text embeddings

Instead of looking only at key words, text embeddings can be used to analyze semantic similarities to other words. This enables us to identify the class of a text using words that are not part of the lexicon.

Text embedding methods create an embedding for each document, which represents the document as a real vector of some fixed dimension q. They are created using the word embeddings of all words in the current document and can be interpreted as an "average" word embedding. We thus interpret the text embedding of a lexicon half as an average embedding of a word of its respective class. Calculating the distance between the embedding of a document in the corpus and the embedding of a lexicon half is used as a measure of how similar a given document is to a theoretical document that is the perfect representation of that class.

As an alternative to the approach mentioned above, we also looked at the average distance of a document's text embedding to all word embeddings of the sentiment words that appear in the document itself, to analyze only its difference to the parts of the lexicon that are part of the document. However, this yielded a classification rate that is comparable to the traditional lexicon classification itself and does not provide substantial improvement over it.

[1]https://github.com/K-RLange/Lex2Sent

It will thus not be further reported in this paper.

The distance is calculated using the cosine distance

$$\text{cosDist}(a, b) = 1 - \frac{\sum_{i=1}^{q} a_i b_i}{\sqrt{\sum_{i=1}^{q} a_i^2} \cdot \sqrt{\sum_{i=1}^{q} b_i^2}}$$

for two vectors $a = (a_1, \ldots, a_q)^T \in \mathbb{R}^q$ and $b = (b_1, \ldots, b_q)^T \in \mathbb{R}^q$ (Li and Han, 2013).

For our purposes of classifying documents into two classes, let A_d be the cosine distance of a text embedding of a document to the text embedding of the positive half of a sentiment lexicon and B_d be the cosine distance to the negative half. Then, the larger (smaller) the value

$$\mathit{diff}_d = B_d - A_d$$

is for a document d, the more confident the lexicon-based text embedding method is, that this document d in fact belongs to class A (B).

This method can be performed using any text embedding model in combination with any lexicon that enables a binary classification task. In this analysis, we choose Doc2Vec (Le and Mikolov, 2014b) as the baseline text embedding model and analyze texts for their sentiment.

3.3 Doc2Vec

Doc2Vec (Le and Mikolov, 2014a) is based on the word embedding model Word2Vec (Mikolov et al., 2013), which assigns similar vectors to semantically similar words by minimizing the distance of a word to the words in its context.

Since word embeddings are not sufficient to classify entire documents, the model is extended to text embeddings. A Doc2Vec model, using the Distributed Memory Model approach, uses a CBOW architecture (Mikolov et al., 2013) in which a document itself is considered a context element of each word in the document. The distance of the document vector to each word vector is minimized in each iteration, resulting in a vector that can be interpreted as a mean of each of its words. According to Le and Mikolov (2014a), these text embeddings outperform the arithmetic mean of word embeddings for classification tasks. In this paper, we use the Doc2Vec implementation of the gensim package in Python (Řehůřek and Sojka, 2010).

Formally, we consider D documents and denote by N_d the number of words in document $d \in \{1, \ldots, D\}$. Further, for $i \in \{1, \ldots, N_d\}$, let $w_{i,d}$ be the i-th word in document d and w_{doc} denote the document under consideration. To give larger weight to words that follow up on another than words that are far away from another, the window size is varied during training. For a Doc2Vec model, we denote by K the maximum size of the context window. For every word the effective size is then sampled from $\{1, \ldots, K\}$ and is denoted as $k_{n,d}$. With these windows, the log-likelihood

$$\sum_{n=K}^{N_d - K} \ln\left(p(w_{n,d} | w_{n-k_{n,d},d}, \ldots, w_{n+k_{n,d},d}, w_{\text{doc}})\right)$$

is maximized for the documents $d = 1, \ldots, D$ using stochastic gradient descent. $p(\cdot|\cdot)$ is calculated by the resulting probabilities from a hierarchical softmax (Mikolov et al., 2013).

We also investigated, if the Lex2Sent method would work when using a pre-trained language model, in this case RoBERTa-large (Liu et al., 2019), as the embedding-backend. For this, we used the CLS-vectors of the lexicon halves and the documents to create lexicon-based text embeddings (similar to Mathew et al. (2020b)). These results underperformed compared to Doc2Vec though, as they showed a bias for one of the two classes.

3.4 Text resampling

Word and text embedding models analyze the original text structure to create similar word embeddings for semantically similar words. We assume that lexicon-based text embeddings need an "optimal text structure" to identify the class of the text in the most efficient manner. Suppose a text contains a key word that is a strong indicatior for the classification task at hand and contained within the lexicon used. The location of such key words can be biased by the type of text. For instance, when analyzing reviews for their sentiment, most key words are located in the last third of the text, as this part draws the conclusion to the review. By resampling the text, we relocate the key words evenly within texts. In theory, this enables vocabulary that occurs more often in texts of a specific sentiment that is not part of any sentiment lexicon, such as topic-specific vocabulary, to be used for labeling texts more efficiently while training Doc2Vec.

We leverage resampling procedures proposed by Rieger et al. (2020), who used them to analyze the uncertainty of the Latent Dirichlet Allocation. Instead of analyzing our methods uncertainty, we use these procedures to create optimal text structures and create a bagging effect. For this, we interpret the original text as a bag of words in which words are drawn independently with replacement like observations when creating a bootstrap sample (Efron, 1979) or independently without replacement, resulting in a permutated text. We call these procedures BW (Bootstrap for Words) and BWP (Bootstrap for Word Permutation), respectively. We analyzed additional procedures, such as resampling sentences as a whole or resampling words only within sentences and variations of those, but these generally yielded lower classification rates than the procedure described above.

3.5 Bagging

In this subsection, we describe a technique to aggregate multiple text embeddings for the purpose of unsupervised sentiment analysis. In combination with resampled texts, this can be seen as a bagging method for unsupervised text classification (Breiman, 1996). Every text structure has an effect on the classification of lexicon-based text embeddings, as differing syntax and vocabulary change the resulting embeddings. However, identifying whether the texts already have an "optimal" structure is a difficult task, as this is an abstract concept that is not trivial to formalize. Instead of relying on the original texts' structure, resampling enables the possibility to create an arbitrary number of artificial texts. If we aggregate these text embedding models, they do not have to label a document correctly for one text structure (that is the original text), but instead only have to label a document correctly on average based on multiple differently structured texts. This aggregation also balances out the randomness of generating samples and the negative effect of missing out on a crucial word within documents in one resampling sample, as it will probably appear in other samples.

The aggregation is performed by calculating an average *diff*-vector using B resampling iterations. Let diff_d^b be the d-th element of the *diff*-vector for the b-th lexicon-based text embedding model with $d = 1, \ldots, D$. Then

$$\mathit{diff}_d^{\text{mean}} := \frac{1}{B} \sum_{b=1}^{B} \mathit{diff}_d^b$$

defines the d-th element of the averaged *diff*-vector.

3.6 Algorithm and Implementation

In training, the algorithm iterates over a grid, calculating models for different training epochs, context window sizes and embedding dimensions. For our application, we use a $3\times3\times4$-grid, which turns out to be sufficiently beneficial in application while remaining computationally feasible. The parameters are chosen from an equidistant set over reasonable parameter choices (see Algorithm 1 for the parameter choices). The grid can be adjusted according to the practitioner's problem at hand. For instance, a smaller grid is faster to train, but a larger grid will lead to more robust results. In each iteration, the parameter combination for the Doc2Vec model is chosen from the grid and the corpus is resampled. The resampled documents are sorted ascendingly by their respective absolute lexicon score. Then we train a Doc2Vec model and calculate the *diff* vector for all iterations. The classification task is performed by using the component-wise arithmetic mean of all the 36 *diff*-vectors. The algorithm is described as pseudocode in Algorithm 1.

Given a classifier $x = (x_1, \ldots, x_D) \in \mathbb{R}^D$, the document with the index $d \in \{1, \ldots, D\}$ is labeled

$$\text{label}_d = \begin{cases} \text{class A}, & x_d - t < 0 \\ \text{class B}, & x_d - t > 0\,, \quad t \in \mathbb{R} \\ \text{at random}, & x_d - t = 0 \end{cases}$$

for some threshold $t \in \mathbb{R}$. This might be $t = 0$ or the empirical quantile $t = x_{(p)}$, where p is the estimated proportion of texts of class B based on a-priori knowledge. In the analyses of this paper, we assume to have no a-priori knowledge of the distribution of class labels, so we use $t = 0$.

4 Data sets and lexica

In this section, the two sentiment lexica and three data sets used to evaluate Lex2Sent are described.

4.1 Data sets

The three data sets considered in this paper are chosen to cover texts with distinct features. The iMDb data set consists of a large corpus with long documents and a strong sentiment compared to the other two data sets. The Airline dataset is more than four times smaller and the documents themselves are also shorter. The Amazon data set represents an intermediate case between these two data sets.

The texts are tokenized and stop words as well as punctuation marks and numbers are removed. Lemmatization is performed to generalize words with the same word stem, if the original word from the text does not already appear in the lexicon. The

Algorithm 1 Lex2Sent

```
1: procedure LEX2SENT(TEXTS, THRESHOLD, LEXICON, RESAMPLING)
2:     classifier ← [0] ∗ length(texts)
3:     for (epoch, window, dim) in Grid = ($\{5, 10, 15\}, \{5, 10, 15\}, \{50, 100, 150, 200\}$) do
4:         resampled_texts ← resampling(texts)
5:         sorted_resampled_texts ← sort(resampled_texts, lexicon)
6:         model ← Doc2Vec(sorted_resampled_texts, epoch, window, dim)
7:         emb ← lexicon_based_text_embeddings(model, resampled_texts)
8:         for i in 1:length(emb) do
9:             classifier[i] += emb[i]
10:    return label_by_threshold(non_resampled_texts, classifier, threshold)
```

mentioned methods and stop word list are part of the Python package *nltk* (Bird et al., 2009).

iMDb data set The iMDb data set consists of $50,000$ user reviews of movies from the website `iMDb.com`, provided by Stanford University (Maas et al., 2011). These are split into $25,000$ training and test documents, each containing $12,500$ positive and negative reviews. After preprocessing, each document in the data set is 120.17 words long on average.

Amazon Review data set The Amazon data set is formed from the part of the Amazon Review Data which deals with industrial and scientific products (He and McAuley, 2016). All reviews contain a rating between one and five stars. Reviews with four or five stars are classified as positive and reviews with one or two stars are classified as negative. We removed reviews with a rating of three stars from the data set because the underlying sentiment is neither predominantly negative nor positive. In addition, we filtered out reviews consisting of less than 500 characters. Out of the remaining documents, $52,000$ documents are split into $26,000$ training and $26,000$ test documents, which are formed from $13,000$ positive and $13,000$ negative documents each. The average length of all documents in the training corpus is 85.51 words after preprocessing.

Airline data set The third data set consists of $11,541$ tweets regarding US airlines and was downloaded from Kaggle (Crowdflower, 2015). The tweets are categorized into positive or negative tweets – 3099 neutral tweets are deleted to be able to use the data set for a two-label-case. We split this data set in half into a training and test set. The training set ultimately contains 5570 documents. On average, each document of the training set contains 10.60 words after preprocessing. In comparison to the other two data sets, where the labels are evenly split, in the Airline data set only 1386 and thus 24.02% of the documents are labeled positive.

4.2 Lexica

To demonstrate that the performance is not dependent on the lexicon chosen as a base, we show the performance for three lexica: The Opinion Lexicon (Hu and Liu, 2004) is used to represent as a review-specific sentiment lexicon, while the WKWSCI lexicon (Khoo and Johnkhan, 2018) is chosen as multiple-purpose lexicon. The Loughran-McDonald (Loughran and McDonald, 2010) lexicon was designed for economic texts and not for reviews, hence it represents the case in which a lexicon is used in a sub optimal domain. To make sure that Lex2Sent not only outperforms these two lexica, we also observed the classification rate when using VADER (Hutto and Gilbert, 2014) or Afinn (Nielsen, 2011) lexicon in the traditional way and compare these results to the one of Lex2Sent in Section 5.3.

We added four amplifiers and ten negations to improve the classification. If an amplifier occurs before a key word, its value is doubled and if a negation occurs, it is multiplied by -0.5. For traditional lexicon methods, the classifier is created by summing up the values of all words within a text.

5 Evaluation

The classification rates of Lex2Sent in this section are determined by evaluating 50 executions to observe the method's randomness and to get a metric for the average performance.

Table 1 displays the average classification rates of a WKWSCI-based Lex2Sent and the classification rate of the best performing sentiment lexicon for each data set, split by the classification-

Table 1: Average classification rate in percent of a WKWSCI-based Lex2Sent in comparison to the best lexicon method (in brackets), split into whether the fixed or proportion threshold is used

	WKWSCI-based Lex2Sent		Lexicon with the highest classification rate	
threshold	by proportion	0	by proportion	0
iMDb	80.93	80.01	76.82 (TextBlob)	73.32 (Opinion Lexicon)
Amazon	77.08	76.83	71.91 (VADER)	69.28 (Opinion Lexicon)
Airline	79.11	72.42	82.05 (VADER)	68.33 (Opinion Lexicon)

threshold used. The WKWSCI-lexicon is chosen as a basis for Lex2Sent as it is a multiple-purpose lexicon. Lex2Sent outperforms every of the 6 observed lexica on all three data sets when using the threshold 0, as it would usually be done in an fully unsupervised setting without a-priori knowledge. It also outperforms the lexica in two out of three cases in which the exact proportion of positive to negative documents is assumed to be known. Here it is only outperformed by VADER on the Airline data set, which is likely because this data set consists of short documents which do not give the Doc2Vec models much context to train on per document.

While Lex2Sent outperforms these lexica, it does not outperform Chat-GPT. Laskar et al. (2023) report that GPT-3.5 (`text-davinci-003`) reaches an 91.9% classification rate on the iMDb data set. While it is not known, if GPT-3.5 has seen this data set and its labels during training and it thus might have an unfair advantage by knowing the correct results (Li and Flanigan, 2024), due to its generally high performance on unsupervised classification tasks, we can assume that it will outperform Lex2Sent, at least on most data sets. Lex2Sent does yield the advantage of not requiring financial backing to analyze large data sets though. Only a CPU is needed.

5.1 Different resampling procedures

In this section, we investigate the effect of different resampling procedures on the performance of Lex2Sent. We examine the results of a WKWSCI-based Lex2Sent using either one of the resampling procedures defined in Section 3.4 or no resampling at all for the iMDb data set. Additionally we investigate the classification rate when using texts sorted by their absolute lexicon value (key words grouped at the end of a text). This serves as an ablation analysis to distinguish the effects of resampled, natural and sub optimal text structures (sorted texts).

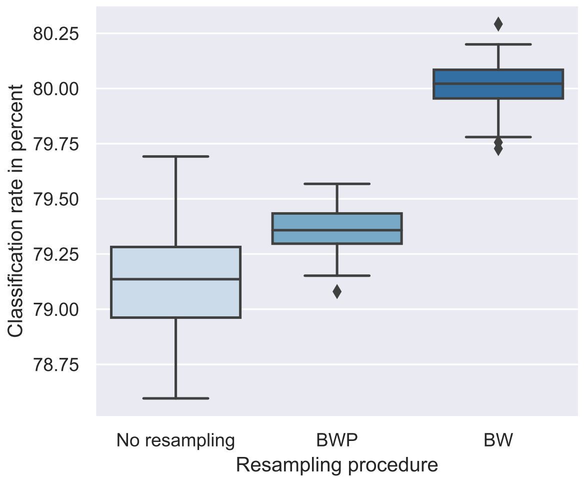

Figure 1: Results of the WKWSCI-based Lex2Sent on the iMDb data set for different resampling procedures

In comparison to the classification rates displayed as boxplots in Figure 1, this subotimal text structure results in a strongly decreased classification rate of 71.00%, which is in line with our interpretation of Section 3.4. The bagging-effect is visible for both procedures, as using either results in higher classification rates for the iMDb data set, with BW yielding the best performance. The method's stability is also increased, as the classification rates are more consistent, which can be seen by comparing the size of the respective box plots. Similar results (not reported) also occur for the other two data sets. For the rest of this paper, all further results are thus reported for Lex2Sent using BW resampling.

5.2 Evaluation on smaller corpora

As Lex2Sent requires training to accurately represent words with embeddings, it is important to determine how large a corpus needs to be for it to provide sufficient results. To analyze this, we evaluate Lex2Sent for subsamples of each data set. These include 10%, 25% or 50% of the original documents. The results of 50 repetitions are displayed in Table 2. The classification rates decrease for smaller corpora except for the Airline data set, in which it

Table 2: Average classification rates in percent of a WKWSCI-based Lex2Sent on subsets of the original data sets for the fixed threshold 0

subsample size	100%	50%	25%	10%
iMDb	80.01	79.73	79.43	78.88
Amazon	76.83	75.71	73.79	68.86
Airline	72.42	72.73	69.74	46.21

Table 3: Average classification rates in percent of Lex2Sent with a WKWSCI-, Loughran McDonald- or Opinion Lexicon-base for the fixed threshold 0, compared to the rates of the traditional lexicon method on the same lexicon

	WKWSCI		Opinion Lexicon		Loughran McDonald	
	Lex2Sent	lexicon	Lex2Sent	lexicon	Lex2Sent	lexicon
iMDb	80.01	70.10	78.43	73.37	70.73	61.22
Amazon	76.83	65.15	77.68	69.28	69.27	61.32
Airline	72.42	63.29	71.96	68.33	72.06	53.18

is slightly higher when examining only 50% of the data set. On the iMDb data set, Lex2Sent outperforms all lexica, even when using just 10% of all documents. On the Airline and Amazon data sets, the classification rate of Lex2Sent decreases to a larger extend for smaller subcorpora. This is likely caused by the short documents in these data set and indicates that it is meaningful to use Lex2Sent on smaller data sets if the documents themselves are long enough to train accurate embeddings.

5.3 Different lexicon-bases for Lex2Sent

So far, we focused on the WKWSCI-based Lex2Sent. In this section, we evaluate, how sensitive Lex2Sent is regarding its lexicon-base and if it improves the classification rate of other lexica as well. For this we compare it to Lex2Sent models based on the Opinion lexicon as well as the Loughran-McDonald lexicon. The average classification rates are displayed in Table 3. Lex2Sent improves the rates of all three lexica on all data sets. While WKWSCI is a general-purpose lexicon, the Opinion Lexicon is designed to analyze customer reviews. This specialization also affects Lex2Sent, as the Opinion Lexicon-based Lex2Sent outperforms every lexicon on every data set as well as the WKWSCI-based Lex2Sent on the Amazon data set, which consists of product reviews. Similarly, we see that Lex2Sent can improve the performance of a lexicon designed for a different domain, as it increases the classficiation rate for the Loughran-McDonald lexicon by at least 7.95 percentage points on all data sets. We recommend to use a general-purpose lexicon like WKWSCI or a lexicon with is domain-adapted to the data set under consideration as a lexicon base for Lex2Sent.

5.4 Lex2Sent as an initial fit

While Lex2Sent is designed for a low hardware resource environment without a GPU, it can still be benefitial to use it in combination with larger, pre-trained models like RoBERTa. To demonstrate this, we use Lex2Sent's beneficial property of displaying a degree of certainty in its results based on how high or low the value of $diff_d^{\text{mean}}$ is for $d = 1, \dots, D$. To create data set for our RoBERTa model to fine-tune on, we therefore only use 10% of the data set: the 5% documents that have the highest and 5% that have the lowest values of our training data set. We fine-tune this version of RoBERTa in 30 epochs using LoRA (Hu et al., 2021) with $r = 8$ and thus 1,838,082 trainable parameters.

To evaluate this approach, we use the iMDb data set, as it contains both a training data set for Lex2Sent to train and RoBERTa to fine-tune on and a test data set for out-of-sample observations that can be classified by RoBERTa. We repeated this procedure five times. On average, our fine-tuned model classified 85.47% of all test documents correctly. While this does not match GPT's classification rate, it does yield the advantage of being cost-efficient. This indicates that Lex2Sent can make for a good initial fit for an active learning approach. Starting from this classification rate, a human-in-the-loop style annotation might take place to improve the classification further.

6 Conclusion

Text classification is commonly performed in a supervised manner using a hand-labeled data set. Unsupervised classification can help when there is no such annotated data set available. This paper proposes the Lex2Sent model, which steers an intermediate course between learning-based and deterministic approaches to create an unsupervised classification, which can be created in a low hardware resource environment without access to a GPU. A binary lexicon is used as a replacement for the missing information that is usually represented by the annotations. The performance of this method is increased by aggregating the results from resampled data sets, which can be seen as a bagging effect.

Lex2Sent yields higher classification rates than all six analyzed sentiment lexica on all three data sets under study, no matter the lexicon-base. Our findings indicate that this might be caused by classifying documents in a more balanced way compared to traditional lexicon methods. Despite being a learning-based approach, the Lex2Sent method shows higher classification rates than traditional lexica on smaller data sets.

Ethical Considerations

While our model requires calculating multiple Doc2Vec models for a single analysis, we modified our model specifications and the number of executions to keep the computational budget manageable in the context of climate change (Strubell et al., 2019). Hence, we perform 50 executions in all of our experiments to ensure that the results are not affected by outliers, but the computational budget remains within reasonable boundaries. Our choice of using the fixed grid with 36 parameter combinations is also caused by this goal. Using this grid, each model finished training in less than two hours.

Limitations

While Lex2Sent improves the classification rate of lexica, it is not capable of reaching the classification rates of models like GPT, but should be seen as a much less resource intensive alternative for the specific task of binary text classification.

Lex2Sent's architecture is independent of the type of binary classification task at hand, so it should work similarly well for other classification tasks given suitable lexica. This is however a theoretical assumption, as we have tested Lex2Sent's capabilities for sentiment analysis specifically.

Lex2Sent has been designed for a two-label-case. To use it in a ordinally scaled multi-label-case, we would need to create multiple thresholds that determines the predicted class, instead of just one. This yields new challenges, as we can not heuristically choose the threshold as 0 like in a binary classification task.

While Lex2Sent's architecture does not depend on the language of the documents or the lexica, it should theoretically perform just as well in low resource languages without needing large training data sets like sophisticated language models. We have not tested this hypothesis though.

Acknowledgments

This paper is part of a project of the Dortmund Center for data-based Media Analysis (DoCMA) at TU Dortmund University. The work was supported by the Mercator Research Center Ruhr (MERCUR) with project number Pe-2019-0044. In addition, the authors gratefully acknowledge the computing time provided on the Linux HPC cluster at TU Dortmund University (LiDO3), partially funded in the course of the Large-Scale Equipment Initiative by the German Research Foundation (DFG) as project 271512359.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *arXiv:2005.14165*.

Crowdflower. 2015. Twitter US airline sentiment.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Bradley Efron. 1979. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26.

Ruining He and Julian J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW '16: Proceedings of the 25th International Conference on World Wide Web*, pages 507–517.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv:2106.09685*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Clayton Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 8, pages 216–225.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. *arXiv:2401.04088*.

Christopher SG Khoo and Sathik Basha Johnkhan. 2018. Lexicon-based sentiment analysis: Comparative evaluation of six sentiment lexicons. *Journal of Information Science*, 44(4):491–511.

Kai-Robin Lange and Carsten Jentsch. 2023. SpeakGer: A meta-data enriched speech corpus of German state and federal parliaments. In *Proceedings of the 3rd Workshop on Computational Linguistics for the Political and Social Sciences*, pages 19–28.

Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Huang. 2023. A Systematic Study and Comprehensive Evaluation of ChatGPT on Benchmark Datasets. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 431–469.

Quoc Le and Tomas Mikolov. 2014a. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196.

Quoc V. Le and Tomas Mikolov. 2014b. Distributed representations of sentences and documents. *arXiv:1405.4053*. Version: 2.

Baoli Li and Liping Han. 2013. Distance weighted cosine similarity measure for text classification. In *Intelligent Data Engineering and Automated Learning – IDEAL 2013*, Lecture Notes in Computer Science, pages 611–618.

Changmao Li and Jeffrey Flanigan. 2024. Task Contamination: Language Models May Not Be Few-Shot Anymore. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):18471–18480.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. 2024. Jamba: A Hybrid Transformer-Mamba Language Model. *arXiv:2403.19887*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692*.

Tim Loughran and Bill McDonald. 2010. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *SSRN Scholarly Paper*, (ID 1331573).

Edward Ma. 2019. Nlp augmentation. `https://github.com/makcedward/nlpaug`.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Binny Mathew, Sandipan Sikdar, Florian Lemmerich, and Markus Strohmaier. 2020a. The polar framework: Polar opposites enable interpretability of pre-trained word embeddings. In *Proceedings of The Web Conference 2020*, WWW '20, page 1548–1558.

Binny Mathew, Sandipan Sikdar, Florian Lemmerich, and Markus Strohmaier. 2020b. The polar framework: Polar opposites enable interpretability of pre-trained word embeddings. In *Proceedings of The Web Conference 2020*, pages 1548–1558.

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

Finn Årup Nielsen. 2011. A new ANEW: evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages*, volume 718 of *CEUR Workshop Proceedings*, pages 93–98.

Christian Rauh. 2018. Validating a sentiment dictionary for German political language—a workbench note. *Journal of Information Technology & Politics*, 15(4):319–343.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Jonas Rieger, Carsten Jentsch, and Jörg Rahnenführer. 2020. Assessing the uncertainty of the text generating process using topic models. In *ECML PKDD 2020 Workshops*, pages 385–396. Springer International Publishing.

Jonas Rieger, Kostiantyn Yanchenko, Mattes Ruckdeschel, Gerret von Nordheim, Katharina Kleinen-von Königslöw, and Gregor Wiedemann. 2024. Few-shot learning for automated content analysis: Efficient coding of arguments and claims in the debate on arms deliveries to Ukraine. *SCM Studies in Communication and Media*, 13(1):72–100.

Timo Schick and Hinrich Schütze. 2021. Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference. *arXiv:2001.07676*.

Bonggun Shin, Timothy Lee, and Jinho D. Choi. 2017. Lexicon integrated CNN models with attention for sentiment analysis. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 149–158.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650.

Alaa Tharwat and Wolfram Schenck. 2023. A Survey on Active Learning: State-of-the-Art, Practical Challenges and Research Directions. *Mathematics*, 11(4):820.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv:2302.13971*.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2020. Unsupervised data augmentation for consistency training. *arXiv:1904.12848*. Version: 6.