Lex2Sent: A bagging approach to unsupervised sentiment analysis

Anonymous ACL submission

Abstract

Unsupervised sentiment analysis is traditionally performed by counting those words in a text that are stored in a sentiment lexicon and then assigning a label depending on the proportion of positive and negative words registered. While these "counting" methods are considered to be beneficial as they rate a text deterministically, their accuracy decreases when the analyzed texts are short or the vocabulary differs from what the lexicon considers default. The model proposed in this paper, called Lex2Sent, is an unsupervised sentiment analysis method to improve the classification of sentiment lexicon methods. For this purpose, a Doc2Vec-model is trained to determine the distances between document embeddings and the embeddings of the positive and negative part of a sentiment lexicon. These distances are then evaluated for multiple executions of Doc2Vec on resampled documents and are averaged to perform the classification task. For three benchmark datasets considered in this paper, the proposed Lex2Sent outperforms every evaluated lexicon, including state-of-the-art lexica like VADER or the Opinion Lexicon in terms of accuracy.

1 Introduction

Most commonly, sentiment analysis is performed in a supervised manner by using a previously labeled dataset to train a learning-based model to predict the sentiment of unlabeled documents. When a labeled dataset is not available, an unsupervised labeling approach is useful to either create a training set for supervised models or to label the texts right away. Unsupervised labeling is usually performed by counting words of a specific sentiment in a text, which are part of a sentiment lexicon.

This paper focuses on lexicon-based text embeddings, an unsupervised labeling approach created by combining learning-based methods with sentiment lexica. We use text embedding models to estimate the similarity between a document and both the positive and negative part of a given lexicon by measuring the distance between their embeddings. These distances are calculated for multiple resampled corpora and are averaged to achieve a bagging-effect. As no labeled texts are available to tune the models parameters, we average models with different parameters during the resampling iterations instead. To demonstrate that the results are generalizable, we compare them to the ones of six traditional lexicon methods on three datasets with distinct characteristics.

The rest of this paper is structured as follows. In Section 2, similar approaches to (unsupervised) sentiment analysis and methods using comparable techniques are presented. Section 3 introduces the concept of Lex2Sent by describing the Doc2Vec-model, the unsupervised labeling approach used and various resampling procedures. The datasets and lexica used are specified in Section 4. In Section 5, the classification rates of Lex2Sent are compared to traditional lexicon methods. In Section 6, we conclude and give an outlook to further research.

2 Related Work

Text embedding methods create vector representations of each word in a corpus, which are then averaged to create a vector representation of each document. These text representations can be used for supervised sentiment analysis by interpreting them as feature vectors for a regression or classification model. Such methods include FastText (Bojanowski et al., 2017, Joulin et al., 2016) and BERT (Devlin et al., 2019).

Mathew et al. (2020) use the distances between embeddings of a document and polar opposite words to create a new embedding and perform supervised sentiment analysis by training a classifier on these new embeddings. Wójcik (2020) tries to combine word embeddings and K-means clustering, but this clustering approach also needs manual work to classify which cluster can be considered positive or negative and the cluster centroids are not guaranteed to accurately represent one sentiment. In contrast, our Lex2Sent method uses text embeddings to create appropriate embeddings both for documents as well as for positive and negative sentiment. It also does not rely on a labeled training set and works in an unsupervised manner.

Some sentiment lexicon methods like VADER (Hutto and Gilbert, 2014) or TextBlob (Loria, 2020) contain a list of negations and amplifiers in addition to the lists of sentiment words. Lex2Sent applies this concept to text embeddings, so that a negated word can be interpreted independently from its non-negated form.

The proposed Lex2Sent method uses resampled texts to amplify its performance. One approach to resample texts is presented by Xie et al. (2020), who proposed to resample words with a probability based on their tfidf-score or by translating the original document into another language and translating it back to the original language. Xie et al. (2020) use these resampling procedures to broaden the training set for supervised sentiment analysis. Such resampling procedures do however either change the vocabulary by back-translation or resample the document dependently from other documents due to the tfidf-scoring. The resulting resampled documents inaccurately represent the original documents, which might be counter-productive for unsupervised analysis. The resampling procedures in this paper are instead based on those of Rieger et al. (2020), as these resample the texts independently from another and do not add new words to the vocabulary.

3 Lex2Sent

In this section, we propose a bagging model for unsupervised sentiment analysis named Lex2Sent, which is implemented in Python (Van Rossum and Drake, 2009).

3.1 Lexica

To perform unsupervised sentiment analysis, sentiment lexica are necessary to interpret the words in a text without the need for previously labeled documents of a similar corpus.

A sentiment lexica assigns a value from an interval $[-s, s]$ to positive and negative words, $s \in \mathbb{R}$, while assigning the value 0 to all neutral words. It assigns positive values to positive words and negative values to negative words. We modify such a lexicon to consist of two halves: a positive and a negative half. These halves are defined as lists of words in a way that each positive or negative word occurs exactly once in its respective half. This enables the use of lexicon-based text embeddings and Lex2Sent.

Amplifiers and negations can improve the performance by affecting the value of the following word. To apply this concept to text embeddings, we merge negations with the following words during preprocessing. The term “not bad” is thus changed to “negbad” and can be interpreted independently from the word “bad”.

3.2 Lexicon-based text embeddings (lbte)

Instead of looking only at sentiment words themselves, text embeddings can be used to analyze semantic similarities to other words. This enables us to classify texts 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 sentiment lexicon half as an average embedding of a positive or negative word. 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 completely positive or negative document.

The distance is calculated using 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).

Let pos_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 neg_d be the cosine distance to the negative half. Then, the larger (smaller) the value

$$diff_d = neg_d - pos_d$$

is for a document d, the more confident the lexicon-based text embedding method is, that this document d is in fact positive (negative).

This method can be performed using any text embedding model in combination with any sentiment lexicon. In this paper, Doc2Vec is chosen as the baseline text embedding model.

3.3 Doc2Vec

Doc2Vec (Le and Mikolov, 2014) is based on the word embedding model Word2Vec (Mikolov et al., 2013a), 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 for classifying 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., 2013a) 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 (2014), 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 $d \in \{1, \dots, D\}$. Further, for $i \in \{1, \dots, N_d\}$, 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 to words that are far away from another, the window size is varied. For a Doc2Vec model, we choose the maximum size the context window can reach and denote it by K. The effective size is then sampled from $\{1, \dots, K\}$ for every word and is denoted as $k_{n,d}$. With these windows, the negative log-likelihood

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

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

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 sentiment of the text in the most efficient manner. Text resampling can help to provide such a text structure by distributing sentiment words across a text. For instance, suppose a positive review is to be analyzed in which most sentiment words are located in the last third of the text, as this part draws the conclusion to the review. By relocating the sentiment words, every word of the document has the same chance to be trained to be similar to sentiment words. 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.

The resampling used in this paper is based on the work of Rieger et al. (2020), who used resampling procedures to analyze the statistical uncertainty of the topic modeling method Latent Dirichlet Allocation. We present the results of three resampling procedures.

- **B**Word: The original text is interpreted as a Bag of Words, in which words are drawn independently with replacement like observations when creating a bootstrap sample (Efron, 1979). The advantage of the method is the high variation of possible texts – for a text length of n_d words, $\binom{2n_d-1}{n_d}$ different variations of the text are possible (if there are no duplicate words in the original text).
- **B**Word**P**ermutation: The words within the text are relocated by permuting them. Thus, the new resampled text has exactly the same vocabulary and length as the original text.
- **B**Neighbor**F**orward: For every word in the original document, the three words following it are added to a context list of that word. Then, to create a resampled text, a word is sampled from the context list of the previously sampled word. This way the "direction" as well as the original context of the text is preserved.

We analyzed additional resampling procedures, such as ones resampling sentences as a whole or resampling words only within sentences and variations of those, but these generally yielded lower accuracies than the three procedures described above.

3.5 Bagging

In this subsection, we describe a technique to combine 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 sentiment analysis (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. We cannot tell if the texts already have an "optimal" structure, as this is a very 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 combine these text embedding models, the text embedding models 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 judging from multiple differently structured texts. This averaging 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 averaging is performed by calculating an average *diff*-vector using B resampling iterations. Let diff_i^b be the i-th element of the *diff*-vector for the b-th lexicon-based text embedding model. Then

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

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

3.6 Algorithm and Implementation

We propose the lexicon-based Lex2Sent method, which uses the following steps. In training, the algorithm iterates over a grid, calculating models for different training epochs, context window sizes and embedding dimensions. For our application, use a 3×3×4-grid, which turns out to be sufficiently beneficial in application while remaining computationally feasible. However, the size of the grid can be modified, to make the method more consistent or faster to train. In each iteration, the current parameter-combination for the Doc2Vec model is chosen from the grid and the corpus is resampled using a resampling procedure. Due to the high variety of possible texts, we use BWord as the default resampling procedure. The resampled documents are sorted ascendingly by their respective absolute lexicon score using the lexicon Lex2Sent is based on. 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.

Algorithm 1 Lex2Sent

```
 1: procedure LEX2SENT(OLD_TEXTS,
 2:                    THRESH, LEXICON,
 3:                    RESAMPLE)
 4:     classifier ← [0] ∗ length(old_texts)
 5:     for (epoch, window, dim) in Grid do
 6:         texts ← resample(old_texts)
 7:         sorted_texts ← sort(texts, lexicon)
 8:         model ← D2V(sorted_texts, epoch,
 9:                     window, dim)
10:         temp ← lbte(model, texts)
11:         for i in 1:length(temp) do
12:             classifier[i] + = temp[i]
13:     return classify(old_texts, classifier,
14:                     thresh)
```

4 Datasets and lexica

In this section, the six sentiment lexica and three datasets used to evaluate Lex2Sent are defined.

4.1 Datasets

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

Preprocessing The texts are tokenized and stop words as well as punctuation marks and numbers are removed. When removing stop words, care is taken not to remove sentiment words, amplifiers or negations to not affect the usage of sentiment lexica. Lemmatization is performed to generalize words with the same word stem. The mentioned methods and stop word list are part of the Python package *nltk* (Bird et al., 2009).

To enable the analysis of negations for text embeddings, we add the prefix "neg-" to a word if the previous word is a negation. We do this in both the texts and the lexicon-bases for the lexicon-based text embedding. For instance, the term "not bad" is turned into "negbad" and is interpreted as a positive term by sentiment lexica and text embeddings.

VADER and TextBlob can be evaluated on a non-tokenized text, as they can preprocess texts themselves prior to analyzing the sentiment. The non-processed texts are therefore stored to be evaluated by VADER and TextBlob.

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

Amazon Review-dataset The Amazon-dataset 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 dataset 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-dataset The third dataset 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 dataset for a two-label-case. We split this dataset 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 datasets, where the labels are evenly split, in the airline dataset only $1,386$ and thus 24.02% of the documents are labeled positive.

4.2 Lexica

To evaluate the performance of Lex2Sent, we compare its results to those of six sentiment lexica. VADER (Hutto and Gilbert, 2014), Afinn (Nielsen, 2011) and the Opinion Lexicon (Hu and Liu, 2004) can be seen as state-of-the-art lexica for a two-label-case in terms of their accuracy (Ribeiro et al., 2016). TextBlob (Loria, 2020) and the WKWSCI lexicon (Khoo and Johnkhan, 2018) are used as additional multiple-purpose lexica. The Loughran McDonald lexicon (Loughran and McDonald, 2010) is designed to analyze financial texts and is considered to investigate the effect of Lex2Sent when using a base that is sub-optimal for the dataset used.

The lexicon methods VADER and TextBlob already consider negations and amplifiers when analyzing texts. For the Loughran McDonald- , Afinn-, WKWSCI- and Opinion Lexicon, we added four amplifiers and ten negations to improve the classification. For these four lexica, if an amplifier occurs before a sentiment word, its value is doubled and if a negation occurs, it is multiplied by -0.5. The classifier for a text is created by summing up the values of all words within it.

4.3 Classification threshold

As we assume that no labeled training set is available and therefore no class probability can be predicted using learning-based models, the classification is performed by heuristic means of counting words in lexica or measuring distances estimated by a text embedding model. By transforming these metrics into a classifier, we classify by using a threshold which splits the documents into a positive and a negative class.

While the threshold of 0 is a logical choice, some methods additionally create an area of uncertainty around 0, in which documents are labeled neutral. VADER for instance only classifies a text as positive or negative, if it is assigned a value ("compound score") of larger than 0.05 or less than -0.05. While this provides a decent approach when looking at a three-label-case (Ribeiro et al., 2016), possibly ignoring thousands of documents in a two-label-case is not an optimal strategy. Alternatively, we can use quantiles of the classifier as a threshold. Knowing that 50% of the analyzed documents are negative and vice versa, we can make sure to classify exactly half of the documents as negative or positive by using the 50% quantile of the classifier as the classification threshold. This is however only a theoretical and practically not feasible option, as the quantile is generally unknown in an unsupervised setting.

We compare the results of using either a quantile threshold or the fixed threshold 0. We interpret the difference between both thresholds as a measure

Table 1: Average accuracy of the best lexicon method and 50 WKWSCI-based Lex2Sent executions on the iMDb, Amazon and Airline-dataset split into whether the fixed or quantile threshold is used

	Lex2Sent		Best lexicon	
threshold	by quantile	0	by quantile	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)

of the method's bias towards one of both classes. The more the accuracy decreases when using 0 as a threshold, the more the classification favors one of both labels over the other. In contrast to the quantile threshold, the fixed threshold 0 can be used in an unsupervised analysis, as no information about the true labels is needed. It is therefore used to determine the best model.

Formally, 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 as

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

for threshold $t = 0$ or the empirical quantile $t = x_{(p)}$, where p is the proportion of negative texts[1].

The resulting classification is interpreted mainly using its accuracy (or classification rate). Averaging metrics over multiple executions is necessary to make sure that the results are not skewed by outliers. The precision and recall metrics (Tharwat, 2020) are not used, as averaging them could lead to misinterpretation.

5 Evaluation

Table 1 displays the arithmetic mean of the classification rates of 50 WKWSCI-based Lex2Sent executions and the classification rate of the best performing sentiment lexicon for each dataset, split by the classification-threshold used. The WKWSCI-lexicon is chosen as a basis for Lex2Sent as it is a multiple-purpose lexicon.

The best lexicon outperforms the WKWSCI-based Lex2Sent when using the quantile threshold ($\approx 76\%$ negative documents) for the Airline-dataset by 2.94 percentage points. This is possibly due to the fact that the Airline corpus consists of shorter and less documents than both other corpora. The Doc2Vec model requires correctly trained word embeddings, so the number and length of documents is more important than for lexica which do not require any form of training.

When looking at the results for threshold 0, which, in contrast to the quantile threshold, is usable in an unsupervised setting, Lex2Sent yields higher classification rates than all evaluated lexica on all datasets. In contrast to traditional lexicon methods, the accuracies of Lex2Sent are more stable when using a fixed instead of the quantile threshold. The traditional methods show a drop in accuracy by 3.50, 2.59 and 13.72 percentage points in accuracy, when comparing both thresholds.

Discussion To provide a stable classification, the lexicon halves must be balanced in terms of word usage. Otherwise the lexicon could assign a false label, because some words have a sub-optimal weight, interpreting documents as either too positive or negative. For instance, if the goal is to analyze a discussion about the happiness of children, most documents will contain the word "happy" at least once, but it is likely used as a topic word rather than as a reflection of the author's opinion. If the lexicon also contains this word, it favors a positive sentiment. For shorter documents a second problem occurs: when a lexicon does not recognize any of the words used, no classification can be performed. Instead, the document must be ignored, classified as neutral, or the label must be guessed.

When using Lex2Sent, these problems are less apparent. Words included in the lexicon are not needed to classify a text, because semantically similar words in the text are used instead. Also the words are not counted, but a distance is calculated. The classification is thus based on a continuous quantity instead of a discrete one, so there will almost certainly be no documents with a distance value of exactly 0 and need to be classified as neutral or guessed. Using distances likely also grants the advantage of being less reliant on balanced lexica. If a word like "happy" appears in most negative

[1]As a two-label-case is assumed, this classification rule is also used for VADER.

Table 2: Average classification rates of 50 WKWSCI-based Lex2Sent executions on subsets of the original datasets 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%

and positive documents, it will be trained to be a neutral word. As the "average" embeddings of both lexicon halves are used, a single word has less of an impact on the classification.

These interpretations are verified on an examplary execution of the WKWSCI-based Lex2Sent and the Opinion Lexicon on the Airline-dataset. Both executions are chosen as they yield similar classification rates to the average of its method described in Table 1. The lexicon labels a total of $2,721$ documents and thus 47.16% of all documents as positive. In reality, there are only $1,386$ and thus 24.02% positive documents. Lex2Sent labels a total of $2,377$ documents as positive and favors the positive sentiment less than the lexicon. This is just an examplary execution of the lexicon and Lex2Sent, but when comparing the methods, similar results occur for all three datasets.

5.1 Different resampling procedures

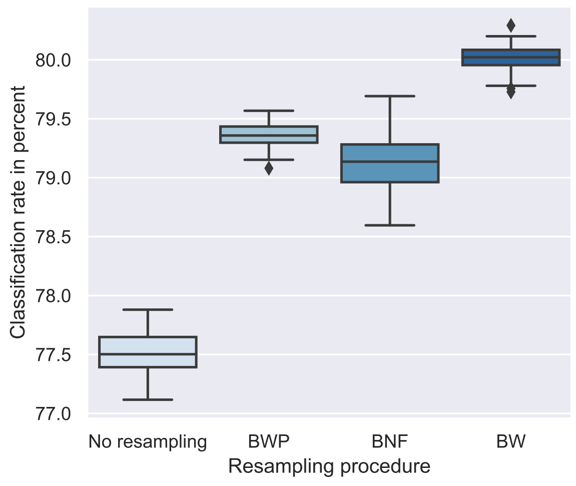

Figure 1: Accuracies of the WKWSCI-based Lex2Sent on the iMDb-dataset for different resampling procedures

In this section we investigate the effect of different resampling procedures on the performance of Lex2Sent. For this, 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. The classification rates for the iMDb-dataset are displayed as boxplots in Figure 1.

Not resampling the texts decreases the accuracy to an average of 77.52%, which is however still 4.20 percentage points higher than highest accuracy of all lexica. The bagging-effect is visible for all three procedures, as using any of these three results in higher classification rates for the iMDb-dataset. Out of these, the BWord procedure originally used yields the highest classification rate on all datasets. Similar results (not reported) also occur for both the Amazon and the Airline dataset.

5.2 Evaluation on smaller corpora

As Lex2Sent combines multiple text embedding models, it is computational expensive, but the accuracy can be transferred to a test dataset. We do this by training a supervised model on the labels resulting from its unsupervised analysis. Using the train-test-split of each dataset, we train a FastText model (with default parameters) on the labels resulting from a WKWSCI-based Lex2Sent with a fixed threshold 0. This model reaches average accuracies of 78.40%, 73.16% and 72.89% respectively, which are each within a range of 3.67 percentage points of the Lex2Sent accuracies and higher than the accuracies resulting from traditional lexica. This preserving effect of a supervised models enables us to label large datasets effectively. We can do this by training Lex2Sent on a sub-corpus and feeding the resulting labels into a supervised sentiment analysis method to classify the rest.

As Doc2Vec requires training to accurately represent words with word embeddings, it is important to determine how large a corpus needs to be for it to outperform traditional lexica. To analyze this, we evaluate Lex2Sent for subsamples of each dataset. These subsamples include 10%, 25% or 50% of the original documents. For each percentage, 50 subsamples are drawn out of the set of documents without replacement for which the WKWSCI-based Lex2Sent is evaluated. The results are displayed in Table 2.

Table 3: Average classification rates of 50 Lex2Sent executions 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%

The classification rates decrease for smaller corpora except for the Airline dataset, in which the classification rate is slightly higher when examining only 50% of the dataset. On the iMDb-dataset, Lex2Sent outperforms all lexica, even when using just 10%, so 2,500 documents. On the Amazon- and Airline datasets however, the classification rate of Lex2Sent decreases to a larger extend for smaller subcorpora. This might be the case because the iMDb-dataset consists of the longest documents – on average, each document is 120 words long, while the average document for the Amazon-dataset is only 86 words long. This indicates that smaller datasets may still be usable for Lex2Sent if the documents themselves are long enough to train accurate word embeddings.

5.3 Different lexicon-bases for Lex2Sent

So far, the main focus of this analysis was the WKWSCI-based Lex2Sent method. In this section we evaluate, how dependent Lex2Sent is on its lexicon-base and if it improves the classification for the Loughran McDonald lexicon and Opinion Lexicon as well. For this, we execute the Lex2Sent method 50 times on each dataset, each for a WKWSCI-, Loughran McDonald- and Opinion Lexicon-basis.

The average classification rates are displayed in Table 3. Lex2Sent improves the classification rates of all three lexica on all three datasets. While the WKWSCI-lexicon is a general-purpose lexicon, the Opinion lexicon is designed to analyze customer reviews. This specialization is also visible for Lex2Sent, as the Opinion Lexicon-based Lex2Sent outperforms every lexica on every dataset as well as the WKWSCI-based Lex2Sent on the Amazon Review dataset.

6 Conclusion

The sentiment of paragraphs or documents is commonly analyzed in a supervised manner using a previously labeled dataset. If there is no labeled dataset available, deterministic sentiment lexicon methods can be used as an alternative to supervised learning-based approaches. This paper proposes a model called Lex2Sent, which is supposed to steer an intermediate course between learning-based and deterministic approaches. While this model uses text embedding models, it does not require labeled data to classify documents. Instead, a sentiment lexicon is used as a replacement for this missing information. The consistency and performance of this method is increased by averaging the results from resampled datasets, which can be seen as a form of bagging. While adjustments to the lexicon-base as well as the size of the grid of Doc2Vec-parameters can be made, this method is designed to be usable in a fully unsupervised manner.

Both the Lex2Sent based on the WKWSCI-lexicon and the one based on the Opinion Lexicon yield higher accuracies than all six analyzed sentiment lexicon methods on all three datasets used in this paper. The findings of this paper indicate that this might be caused by classifying documents in a more balanced way compared to traditional sentiment lexicon methods, as all lexica tend to classify too many documents as positive in the analyzed scenarios. Despite being a learning-based approach, the Lex2Sent method shows higher classification rates than traditional lexica on smaller datasets with training sets consisting of less than 10,000 documents, but partially does not perform as well as traditional sentiment lexica when less than 3,000 documents are available.

To use Lex2Sent in a three-label-case, another classification step needs to be added. Creating deterministic thresholds for neutral documents is not trivial as it is for lexicon methods, such as VADER, as the scale of distances used by Lex2Sent differs for repeated executions. Alternatively an additional analysis to filter out neutral texts can be performed first. This poses the issue of accurately detecting neutral sentiment and relying on the results of two different methods.

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.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *arXiv:1607.04606*.

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

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, Minneapolis, Minnesota. Association for Computational Linguistics.

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

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*, page 507–517. International World Wide Web Conferences Steering Committee.

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. ACM.

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.

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

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. Publisher: SAGE Publications Ltd.

Quoc Le and Tomas Mikolov. 2014. 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, Bejing, China. PMLR.

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. Springer.

Steven Loria. 2020. textblob documentation. V0.16.0.

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

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, Portland, Oregon, USA. Association for Computational Linguistics.

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

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

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.

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, Valletta, Malta. ELRA.

Filipe N Ribeiro, Matheus Araújo, Pollyanna Gonçalves, Marcos André Gonçalves, and Fabrício Benevenuto. 2016. SentiBench - a benchmark comparison of state-of-the-practice sentiment analysis methods. *EPJ Data Science*, 5(1):23.

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, Cham. Springer International Publishing.

Alaa Tharwat. 2020. Classification assessment methods. *Applied Computing and Informatics*, 17(1):168–192. Publisher: Emerald Publishing Limited.

Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Rafał Wójcik. 2020. Unsupervised sentiment analysis.

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.