Utilizing Pre-Trained Word Embeddings to Learn Classification Lexicons with Little Supervision

Frederick Blumenthal
d-fine GmbH
frederick.blumenthal@d-fine.de

Ferdinand Graf
d-fine GmbH
ferdinand.graf@d-fine.de

Abstract

A lot of the decision making in financial institutions, regarding particularly investments and risk management, is data-driven. An important task to effectively gain insights from unstructured text documents is text classification and in particular sentiment analysis. Sentiment lexicons, i.e. lists of words with corresponding sentiment orientations, are a very valuable resource to build strong baseline models for sentiment analysis that are easy to interpret and computationally efficient. We present a novel method to learn classification lexicons from a labeled text corpus that incorporates word similarities in the form of pre-trained word embeddings. We show on two sentiment analysis tasks that utilizing pre-trained word embeddings improves the accuracy over the baseline method. The accuracy improvement is particularly large when labeled data is scarce, which is often the case in the financial domain. Moreover, the new method can be used to generate sensible sentiment scores for words outside the labeled training corpus.

1 Introduction

A vast amount of information in business and especially in the finance area is only available in the form of unstructured text documents. Automatic text analysis algorithms are increasingly being used to effectively and efficiently gain insights from this type of data. A particularly important text analytics task is document classification, i.e. the task to assign a document to a category within a set of pre-defined categories. For example, annual reports, news articles and social media services like twitter provide textual information that can be used in conjunction with structured data to quantify the creditworthiness of a debtor. To give another example, intelligent process automation may require the categorization of documents to determine the process flow. In both cases, sound text classification algorithms help saving costs and efforts.

To tackle the problem of document classification, classical methods combine hand-engineered features, e.g. word-count based features, n-grams, part-of-speech tags or negations features, with a non-linear classification algorithm such as Support Vector Machine (Joachims, 1998). A detailed survey of classical sentiment analysis models, a special case of text classification, has been compiled by Pang et al. (2008) and Liu (2012).

Since the reign of deep learning, various neural network architectures such as convolutional neural networks (CNN) (Kim, 2014; dos Santos and Gatti, 2014), character level CNNs (Zhang et al., 2015), recursive neural networks (Socher et al., 2013), recurrent neural network (RNN) (Wang et al., 2015; Liu et al., 2016) and transformers (Vaswani et al., 2017) have been utilized in text classification models to yield state-of-the-art results.

Recently, a steep performance increase has been achieved by very large pre-trained neural language models such as ELMo (Peters et al., 2018), BERT (Devlin et al., 2018), XLNet (Yang et al., 2019) and more (Howard and Ruder, 2018; Radford et al., 2018; Akbik et al., 2018). These models generate powerful text representations that can be either used as context-aware word embeddings or the models can be directly fine tuned to specific tasks.

One disadvantage of these pre-trained language models, however, is the high demand of memory and computing power, e.g. a sufficiently large GPU to load the large models. In finance, many documents that can be the subject of text classification applications (e.g. annual reports or legislative documents), are very large, so that the computational cost becomes very relevant. Another disadvantage is that because of their complexity, many state-of-the-art deep learning models are hard to interpret and it is very difficult to retrace the model predictions. Model interpretability, however, seems to be particularly important for many financial institutions and interpretable models with transparent features are often favored over more complex models even if the complex models are more accurate.

A powerful resource for building *interpretable* text classification models are classification lexicons and in particular sentiment lexicons. A sentiment lexicon is a list of words (or n-grams) where each word is assigned a sentiment orientation. The sentiment orientation can be binary, i.e. each word in the lexicon is labeled as *positive* or *negative*, or continuous where a continuous sentiment score is assigned to the words (e.g. in the interval [-1, 1]). More generally, a classification lexicon is a list of words where each word is assigned a vector with one score for each class.

Sentiment lexicons have been an integral part of many classical sentiment analysis classifiers (Mohammad et al., 2013; Vo and Zhang, 2015). Approaches based on sentiment lexicons seem to be particularly popular in the finance domain (Kearney and Liu, 2014). In addition, it has been shown that even modern neural network models can profit from incorporating sentiment lexicon features (Teng et al., 2016; Qian et al., 2016; Shin et al., 2016). Using classification lexicon features can be thought of as a way of inducing external information that has been learned from different data sets or compiled by experts.

Three approaches to sentiment lexicon generation are usually distinguished in the literature, namely the manual approach, the dictionary-based approach and the corpus-based approach, see for example (Liu, 2012, Chapter 6). A popular finance specific lexicon has been compiled by Loughran and McDonald (2011) from 10-K fillings , but see also the General Inquirer (Stone et al., 1962) and the Subjectivity Lexicon (Wilson et al., 2005).

Fairly recently, models have been designed to generate sentiment lexicons from a labeled text corpus. In many cases distant supervision approaches are employed to generate large amounts of labeled data. For example, Mohammad and Turney (2013) compiled a large twitter corpus where noisy labels are inferred from emoticons and hashtags. Count-based methods such as pointwise mutual information (PMI) generate sentiment scores for words based on their frequency in positive and negative training sentences (Mohammad and Turney, 2013; Kiritchenko et al., 2014).

A more direct approach to learn sentiment lexicons from labeled corpora is to use supervised machine learning. The basic idea is to design a text classification model that contains a parametrized mapping from word token to sentiment score and an aggregation of word-level sentiment scores to document scores. The parametrized mapping which yields the sentiment lexicon is learned during training. Severyn and Moschitti (2015) proposed a linear SVM model and showed that the machine learning approach outperforms count-based approaches. A simple linear neural network model has been proposed by Vo and Zhang (2016). A similar model with a slightly more complex neural network architecture is used by Li and Shah (2017). They use data from StockTwits, a social media platform designed for sharing ideas about stocks, which they also use to generate sentiment-specific word embeddings.[1] Pröllochs et al. (2015) design a linear model and add L1 regularization to optimally control the size of the sentiment lexicons.

We see two main challenges for the generation of new domain specific classification lexicons via a pure supervised learning approach.

- The generation of robust classification lexicons requires large amounts of supervised training data. Manual labeling of data is very expensive and a distant (or weak) labeling approach may not be possible for all applications.
- Using small or medium size supervised training data, one may encounter many words at prediction time that are not part of the training corpus.

[1]The objective of sentiment-specific word embeddings, first proposed by Maas et al. (2011), is to map words (or phrases) close to each other if they are both semantically similar and have similar sentiment. A sentiment lexicon could be considered as one-dimensional or two-dimensional word embeddings.

To tackle these problems, we propose a novel supervised method to generate classification lexicons by utilizing unsupervised data in the form of pre-trained word embeddings. This approach allows to build classification lexicons with very small amounts of supervised data. In particular, it allows extending the classification lexicon to words outside the training corpus, namely to all words in the vocabulary of the pre-trained word embedding.

The remainder of this paper is structured as follows. Section 2 gives a short introduction to supervised learning of classification lexicons in general and then introduces the novel model extension to utilize pre-trained word embeddings. We show empirically in Section 3 that the use of pre-trained word embeddings improves prediction accuracy and generates better classification lexicons. The accuracy improvement is particularly large for small training data sets. In addition, we show that the model generates sensible word-level class scores for words that are not part of the training data. For the experiments we use the popular SST-2 sentiment analysis dataset which is part of the GLUE benchmark and a new dataset of manually labeled financial newspaper headlines. In Section 4 we describe how a modification of the proposed method can be applied to hierarchical (multi-level) document classification and supervised sentence highlighting in large documents.

2 Methodology

The goal is to learn a classification lexicon, that is, for a given set of word tokens (or n-grams) $\mathcal{D} = \{x^{(l)}\}_{l=1}^{L}$, the task is to learn a domain specific function $\boldsymbol{s} : \mathcal{D} \to \mathbb{R}^C$ that assigns each token a vector of class scores. The resulting classification lexicon $\mathcal{L}$ is then defined as the set of tuples consisting of tokens $x^{(l)}$ and corresponding C-dimensional class scores $\boldsymbol{s}_l$,

$$\mathcal{L} = \{(x^{(1)}, \boldsymbol{s}_1), \ldots, (x^{(L)}, \boldsymbol{s}_L)\}. \tag{1}$$

In the specific case of sentiment analysis, the function $\boldsymbol{s}$ may be two-dimensional with channels for positive and negative sentiment or higher-dimensional in order to represent fine-grained nuances of sentiment.

For supervised learning of the classification lexicon, a data set with labeled text sentences is used, i.e. a data set $D = \{(t_n, y_n)\}_{n=1}^{N}$ that consists of sentences (or other pieces of text) t_n with corresponding class label $y_n \in \{1, \ldots, C\}$. In this setting, the overall idea is to design a classification model that consists of an elementwise mapping $\boldsymbol{s}$ from word token to word-level class scores and a function f that aggregates the word class scores to sentence-level class probabilities,

$$\boldsymbol{p}(t) = \boldsymbol{f}\left(\boldsymbol{s}(x_1), \boldsymbol{s}(x_2), \ldots, \boldsymbol{s}(x_{|t|})\right), \tag{2}$$

with $\boldsymbol{p} \in [0,1]^C$ and $|t|$ denotes the number of words in sentence t. The objective is to learn the functions $\boldsymbol{s}$ and $\boldsymbol{f}$ such that the model as accurately as possible predicts the sentence class labels of the training data. The learned function $\boldsymbol{s}$ then yields the mapping to generate the classification lexicon.

Note that this is a special case of a more general class of hierarchical (multi-level) text classification models that generate class scores for low-level segments and then aggregate these scores to produce document-level classifications. This is discussed in more detail in Section 4.

In order to assure that the learned function $\boldsymbol{s}$ actually produces sensible word-level class scores, the following two conditions have to be fulfilled.

- The function $\boldsymbol{s}(x)$ that maps a token to a class score must not depend on context, i.e. each word token in the lexicon must be mapped to a unique class score value. If the mapping was context dependent, then a single word might be assigned to multiple class scores.
- The aggregation function $\boldsymbol{f}$ must be designed such that the predicted sentence-level class probabilities have a clear dependence on the word-level class scores. In particular, an increase in a certain word-level class score must ceteris paribus increase the sentence-level probability for this class (more than for any other class). That is, for each sentence t, each class $c' \neq c \in \{1, \ldots, C\}$ and each token $x \in t$,

$$\frac{\partial p_c(t)}{\partial s_c(x)} > \frac{\partial p_{c'}(t)}{\partial s_c(x)}. \tag{3}$$

To design a model instance in this general setting one has to specify the mapping $\boldsymbol{s}(x)$ and the function $\boldsymbol{f}$ from Eq. (2) such that the above conditions are satisfied.

2.1 Baseline

Arguably the simplest instance of the described approach, which we use as our baseline model, is to use as function $\boldsymbol{s}$ a direct mapping and as aggregation function $\boldsymbol{f}$ a simple averaging followed by a softmax function. Very similar models have been proposed in previous works (Severyn and Moschitti, 2015; Pröllochs et al., 2015; Vo and Zhang, 2016).

Representing the word tokens x as one-hot vectors, the direct mapping $\boldsymbol{s}$ from word token to word-level class scores can be formulated as a simple matrix-vector multiplication,

$$\boldsymbol{s}(\boldsymbol{x}) = S\boldsymbol{x}, \tag{4}$$

where S is the class score embedding matrix of dimensionality $C \times L$. The columns of matrix S give the classification lexicon, i.e. the l^{th} column gives the class scores for token $\boldsymbol{x}^l$. The word-level class scores are then averaged to compute sentence level class scores,

$$\boldsymbol{z}(t) = \frac{1}{|t|} \sum_{\boldsymbol{x} \in t} \boldsymbol{s}(\boldsymbol{x}) \tag{5}$$

that are finally normalized to yield probabilities,

$$p_c(t) = \frac{e^{z_c(t)}}{\sum_{c'=1}^{C} e^{z_{c'}(t)}}. \tag{6}$$

The only parameters of the model are the elements of the class score matrix S, that is, the elements of the classification lexicon. To tune the model parameters we minimize the average cross-entropy over the training data,

$$L_{CE}(D|S) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} y_{nc} \log p_c(t_n) + \lambda |S|_1 \tag{7}$$

where $L1$ regularization is added as proposed by Pröllochs et al. (2015). Is is known that $L1$ regularization tends to drive model parameters to zero which in this case reduces the size of the classification vocabulary. This behavior can be desirable because many words in the training data (e.g. stop words) are not expected to carry any sensible class score.

2.2 New approach

In the baseline model, a direct mapping from word token to word-level class score is learned from scratch for every word token. In particular, no prior knowledge about semantic relationships between word tokens is considered in the model. Semantic similarity between words can be captured very well by pre-trained word embeddings such as *word2vec* or *GloVe*. Therefore, we propose a classification lexicon model that is build on top of word embeddings. This way, prior knowledge is induced into the model that has been previously learned from a very large and representative unsupervised corpus. This should be particularly useful when learning a classification lexicon from a small supervised corpus.

For the token-level score function $\boldsymbol{s}$ from Eq. (2) a two-step function is designed that first maps the word token to its word vector and then transforms the word vector to a token class score,

$$\boldsymbol{s}(\boldsymbol{x}) = \bar{\boldsymbol{s}}(\boldsymbol{w}(\boldsymbol{x})) \tag{8}$$

where $\boldsymbol{w}(\boldsymbol{x})$ is the word embedding of token $\boldsymbol{x}$ with dimensionality E. The aggregation function is the same as in the baseline model, that is, the class score of a sentence is modeled as the average over the word scores which are then normalized by a soft-max function, see Eq. (5) and (6).

The function $\bar{\boldsymbol{s}} : \mathbf{R}^E \to \mathbf{R}^C$ is modeled as a multilayer fully connected neural network with ReLU activations,

$$\begin{aligned} \boldsymbol{h}^{(1)} &= \text{ReLU}\left(W^{(1)} \boldsymbol{w}(\boldsymbol{x})\right) \\ \boldsymbol{h}^{(2)} &= \text{ReLU}\left(W^{(2)} \boldsymbol{h}^{(1)}\right) \\ &\vdots \\ \boldsymbol{h}^{(H)} &= \text{ReLU}\left(W^{(H)} \boldsymbol{h}^{(H-1)}\right) \\ \boldsymbol{s} &= W^{(final)} \boldsymbol{h}^{(H)}. \end{aligned} \tag{9}$$

We choose all of the H hidden layers to be of some fixed length I, the word-level class scores $\boldsymbol{s}$ have length C. This gives a total of $I\,(E + (H-1)I + C)$ parameters. A high-level sketch of the classification lexicon model is shown in Figure 1. It should be noted that the same function $\bar{\boldsymbol{s}}$ is applied independently to each word token. This can be efficiently implemented e.g. by a convolutional layer with kernel size 1.

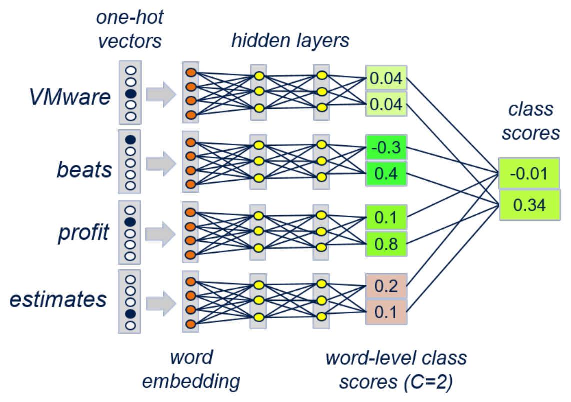

Figure 1: Sketch of the word embeddings based classification lexicon model for a dictionary of $L = 5$ words, a $E = 4$ dimensional word embedding, C=2 classes, $H = 2$ hidden layers with $I = 3$ hidden units. To predict the class probabilities of a piece of text, the word-level class scores are computed from pre-trained word embeddings via a set of of linear transformations followed by rectifiers. The class prediction for the text is computed as the average over word-level class scores.

Since the word embeddings in the model are trained in an unsupervised fashion it is possible that words with very different true class scores are assigned very similar word vectors. Fine-tuning the word embeddings during training could help to separate words with similar pre-trained embedding but different true class scores. However, we decide not to fine-tune the word embeddings during training, because we want to apply the mapping $\bar{s}$ to words that are not part of the training data. Moreover, fine-tuning the word embeddings, which would introduce an additional set of $E \cdot L$ model parameters, did not improve the model accuracy in the experiments.

3 Experiments

The purpose of the proposed classification model is to generate powerful application specific classification lexicons and we want to show that the new model generates better lexicons than the baseline model. To this end, we train both models on two binary sentiment analysis datasets and compare the test set accuracy as a proxy for the classification lexicon quality. Since the new word-embedding based model and the baseline model contain the same aggregation function, any improvement in model predictions must result from the word-level classification scores, i.e. the learned classification lexicons.

The first dataset that we use for the evaluation is the SST-2 dataset (Socher et al., 2013) that contains binary labeled movie reviews. This well-known dataset is publicly available and part of the GLUE benchmark (Wang et al., 2018). The second dataset, which we call *FNHL*, consists of financial news headlines that have been manually labeled by experts. Table 1 shows simple examples from both datasets and Table 2 gives basic dataset statistics. It should be emphasized that the proposed model is not restricted to binary classification problems and could also be applied to multi-class datasets.

FNHL
(+) *French rail network gets three offers for new line*
(-) *Google, Facebook to face tougher EU privacy rules*

SST-2
(+) *the movie exists for its soccer action and its fine acting*
(-) *the plot grinds on with yawn-provoking dullness*

Table 1: Examples from the SST-2 and FNHL datasets.

Dataset	mean($\|t\|$)	N	$\|V\|$	$\|V_{w2v}\|$
SST-2	19	9613	16182	14826
FNHL	10	2792	5885	4664

Table 2: Average sentence length (mean($|t|$)), total dataset size (N), vocabulary size ($|V|$) and vocabulary that is contained in word2vec ($|V_{w2v}|$). Computed on the pre-processed datasets.

Both the baseline and the new model are implemented as neural networks and optimized via the Adam optimizer. For the baseline model dropout regularization is applied to the word level class scores and for the new model dropout is applied before the rectifiers. The new model is implemented with pre-trained word2vec word embeddings. For words that are not contained in word2vec the embedding is set to a vector of zeros. Since the embedding model can always be refined based on an unlabeled domain-specific corpus, one can ensure that the embedding model contains the relevant vocabulary. The SST-2 dataset is provided with a train/dev/test split which is used in our experiments whereas for the FNHL dataset nested cross-validation is used. The dev set is used for early stopping and to evaluate model hyperparameters via grid-search. The optimal hyperparameters are provided in Table 7 in the appendix.

3.1 Model Accuracy

Table 3 shows that the new model outperforms the baseline model on both datasets which means that the new model generates better sentiment lexicons. As an additional experiment we implement the new model with ELMo embeddings which further increases the accuracy on the SST-2 dataset by 3.7%. Since ELMo embeddings are context-dependent this model does not yield a fixed sentiment lexicon but instead yields a mapping from sentence-token pair to sentiment scores.

To put the accuracy of the baseline model and the new classification lexicon model into perspective, we show in Table 3 the accuracy on SST-2 for several GLUE benchmark models as well as recent state-of-the-art models as reported on the official GLUE website, see https://gluebenchmark.com/leaderboard. CBoW denotes an average bag-of-words model using GloVe embeddings, GenSen (Subramanian et al., 2018) denotes the GLUE benchmark sentence representation model with best overall score and InferSent (Conneau et al., 2017) denotes the GLUE benchmark sentence representation model with best SST-2 score. For these models a mapping from sentence representation to class scores was trained. Our new classification lexicon model outperforms the baseline models CBoW and GenSen whereas InferSent achieves slightly better accuracy.

The BiLSTM model with ELMo embeddings and attention (BiLSTM+ELMo+Attn) achieves only 2.6% higher accuracy than NewElmo, i.e. a simple mapping from ELMo to token level class scores. As expected, the popular BERT model and XLNet, the currently best performing model on the SST-2 task, achieve much better accuracy than our proposed classification lexicon model. It should be emphasized, however, that the purpose of the proposed model is not to achieve state-of-the-art accuracy but to generate powerful sentiment lexicons. Therefore, the most relevant result is that the proposed model outperforms the baseline classification lexicon model which shows that the new model generates better sentiment lexicons.

To evaluate the dependency between training set size and model accuracy, the experiments are repeated with subsampled SST-2 training sets, see Figure 2. For small training sets, the new model outperforms the baseline model by a large margin. For example, with 1% of training samples (69 samples) the new model achieves 69% accuracy compared to 54% for the baseline model and with 5% of training samples (346 samples) the new model yields an accuracy of 79% compared to 66% for the baseline model.

	FNHL	SST-2
Baseline	77.4 (75.0, 78.4)	82.5
New	82.8 (82.1, 83.9)	84.1
NewElmo	-	87.8
CBoW	-	80.0
GenSen	-	83.1
InferSent	-	85.1
BiLSTM+ELMo+Attn	-	90.4
BERT	-	94.9
XLNet	-	96.8

Table 3: Accuracy of the baseline and new classification lexicon models. NewElmo denotes the implementation of the new model with ELMo embeddings (which does not yield a lexicon). For comparison, the accuracy on the SST-2 task are shown for the GLUE baseline models CBoW, GenSen, InferSent and BiLSTM+ELMo+Attn as well as the popular BERT model and the currently best performing model XLNet.

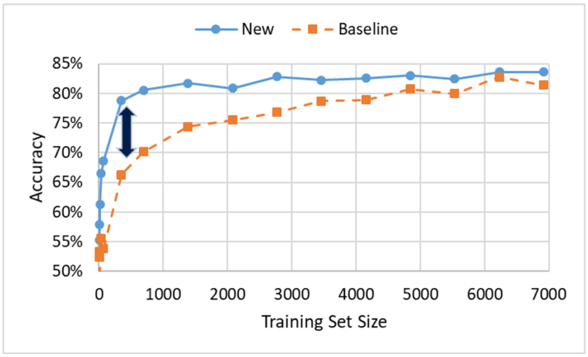

Figure 2: Prediction accuracy on the SST-2 dataset with training set subsampled to different sizes. For small training set sizes the new model significantly outperforms the baseline model.

3.2 Sentiment Lexicons

After a quantitative comparison of the new and baseline classification lexicon models, we now want to take a qualitative look at the generated lexicons. Table 4 shows the tails of the sentiment lexicons as generated by the new model on the FNHL and SST-2 datasets. The well-known domain-specific character of sentiment lexicons is apparent.

Positive words	Negative words
SST-2	
melds, combines, marvelously, enhances, hearts, sublimely, breathtaking, wonderfully, engagingly, supple, winningly, searing, enables, heartwarming, integrates, captures, mesmerizing, infuses, masterly, explores	charmless, ineffective, garbled, misfire, itis, useless, uncreative, dumped, uninspiring, overinflated, unimaginative, unfocused, incoherent, drowned, unambitious, pointless, half-hearted, suffers, faulty, squandering
FNHL	
wins, bt, topping, airshow, turbines, awarded, selected, supercomputer, clinch, debut, paves, beats, tops, inks, secures, buoyed, success, boosted, driverless	violated, violations, falls, lapses, delisted, underreporting, violating, fined, plummet, threatened, misled, sues, fining, drags, infringe, delisting, halts, breaches, fines, censures

Table 4: Example of words in the sentiment lexicons trained on the FNHL and SST-2 datasets using the word-embedding based model.

Table 5 shows the largest word-level sentiment score differences between baseline and new model. Qualitatively, the new model seems to generate more sensible sentiment scores. For the comparison, the two-channel word-level scores are first transformed to a scalar score, $\frac{s_{pos}-s_{neg}}{s_{pos}+s_{neg}}$, and normalized to $[-1, 1]$.

word	Base	New	word	Base	New
ineffective	0.0	-1.0	melds	0.0	0.7
dumped	0.0	-0.9	seagal	-0.6	0.0
garbled	-0.2	-1.0	sweetest	-0.3	0.3
uncreative	-0.1	-0.9	supple	0.0	0.6
itis	-0.2	-1.0	pay	-0.7	-0.1
overinflated	-0.1	-0.9	spry	-0.1	0.4
atrociously	0.0	-0.7	enables	0.0	0.6
uninspiring	-0.2	-0.9	queen	-0.6	-0.1
moldy	0.0	-0.7	convenient	-0.4	0.1
counterproductive	-0.1	-0.7	windtalkers	-0.5	0.0
unambitious	-0.2	-0.8	sheridan	-0.5	0.0
miserably	0.1	-0.6	guess	-0.6	-0.1
knockoff	-0.1	-0.8	dynamic	-0.4	0.2
untalented	-0.1	-0.7	equilibrium	-0.6	0.0

Table 5: Largest differences between sentiment lexicons generated by the baseline and new model.

3.3 Lexicon Extension

By design, the baseline model can only generate word-level class scores for words that are contained in the training corpus. The new model on the other hand learns an application specific mapping from word embedding to word-level class scores. This makes it straight forward to generate class-scores for words outside the training corpus. To evaluate this property we apply the learned mapping (from SST-2 dataset) to a subset of the pre-trained word vectors in word2vec. The word2vec set is filtered to lowercase 1-grams, i.e. phrases are excluded. This leaves a total of 180000 words which is more than 10 times the number of words in the SST-2 training set vocabulary.

Table 6 shows the most positive and most negative sentiment words when applied to the 180000 tokens in word2vec. Most of the words look sensible, which shows that it is possible to generate sentiment scores for words that are not contained in the training corpus. Arguably, this ability to generate scores for unseen words is the reason why the new model significantly outperforms the baseline model on very small training sizes as shown in Figure 2. Of course, the extension of the lexicons also generates poor scores for some words. Qualitatively unplausible words are underlined in Table 6. In general during all sentiment lexicon model evaluations we got the impression that negative words have better quality than positive words.

Positive words
equips, revolutionizing, amazes, reconnects, delighting, soothes, optimizes, prayerfully, backflip, accelerations, empowers, nourishes, maximizes, flyby, centenarians, transfixing, juxtaposes, exhilaratingly, purifies, frugally, caresses, predeceased, glistened, livability, centenarian, policyowners, gratified, securityholders, astound, electrifying, sacraments, equanimity, synchronizes
Negative words
uncompetitive, unproductive, overstocking, misaligned, misconfigured, mistyped, spams, fritz, untargeted, scrapyard, clunked, uninformative, slouching, unworkable, knockoffs, unmarketable, mixup, ineffectively, misdirected, forlornly, misspell, polluter, overleveraged, overwrites, dumper, plagiarized, unemployable, unimpressive, defective, overloaded, flunky, laminitis

Table 6: Words in the word2vec set (filtered for lowercase 1-grams) with most positive and most negative sentiment as generated by the proposed model that has been trained on the SST-2 training set. Most word sentiments are plausible, unplausible words are underlined.

4 Hierarchical Document Classification

In some document classification tasks in the finance domain one deals with very long documents, such as annual reports or legislative documents, that may consist of more than 100 pages. In order to make model predictions more interpretable it would be desirable that the predictions on document level can be retraced to the sentence (or paragraph) level. One advanced approach to achieve this level of locality is to incorporate sentence-level attention in the document-level model, see for example (Yang et al., 2016). For each sentence the attention function indicates how relevant the sentence is for the document-level model prediction. This makes the model predictions more interpretable, i.e. the analyst could better understand the model predictions by looking at the most relevant sentences.

A somewhat simpler approach is to build a model that generates class scores per sentence and then aggregates these scores to document-level class scores. By designing the aggregation such that the document-level scores are in a direct relationship to the sentence-level scores, one can train a joint model for document-level classification that – at the same time – generates sentence-level predictions. This approach is analogous to the classification lexicon model where word-embeddings are replaced by sentence representations. See Figure 3 for a sketch of the model. The sentence representation model is arbitrary and could be for example a pre-trained language model such as BERT or a jointly trained BiLSTM pooling of ELMo embeddings.

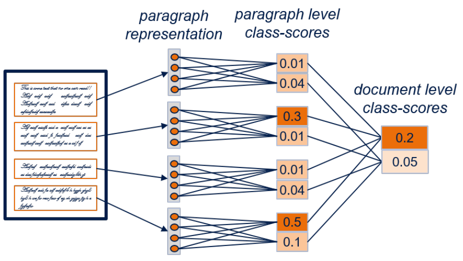

Figure 3: Sketch of a basic architecture for hierarchical document classification. Sentence representations can be computed for example by average pooling all word embeddings in the sentence. Sentence representations are mapped by a parametric function to yield the sentence score for each class. Finally, aggregation (e.g. simple averaging) of the sentence-level class scores yields the document level scores.

The described approach localizes model predictions to the sentence level and thereby makes predictions on large documents interpretable. In addition, the approach can be utilized as a supervised method to highlight important sentences in a document. For example, a financial institution that has to process a large number of annual reports or fund reports can employ such methods to point the analyst to the important parts of the document. In such an application the final document prediction may not be relevant primarily, but the highlighting via sentence level scores is important. Highlighting approaches that we currently see in practice are mostly based on unsupervised text-summarization algorithms such as LexRank (Erkan and Radev, 2004) , which also determines an importance score on sentence-level based on non-parametrical similarity measures and graph-methods, and can also be used in conjunction with our approach.

During our literature review on hierarchical document classification, no model was found that is comparable to the approach described above. However, the general idea to design a joint model for document-level classification that generates sentence-level predictions as a byproduct is not new and has been proposed for example by Yessenalina et al. (2010).

5 Conclusion

This paper presents a novel supervised method to generate classification lexicons that utilizes unsupervised learning in the form of pre-trained word embeddings. The method allows to build classification lexicons, e.g. sentiment lexicons, from very small amounts of labeled data and the model allows to extend the lexicons to words that are not contained in the training corpus. This is very relevant for applications in the financial and compliance area, where labeled data is very sparse and usually very unbalanced. In addition, in these areas cross-institutional data pooling is usually not possible for data protection reasons, and data encryption would render the data useless.

It was shown that using the proposed method with context-dependent word embeddings such as ELMo yields powerful word-level features.[2]

To improve the overall classification lexicon

[2]Implementing the approach with context-dependent word-embeddings yields a context-dependent mapping from words to class scores and thus does not produce a classification lexicon.

model the knowledge distillation approach (Ba and Caruana, 2014; Hinton et al., 2015) could be used where a simple model is trained on the raw predictions of a more complex model. In our case the new classification lexicon model could be trained for example on the class scores (scores before softmax function) of BERT or XLNet. The potential improvements of distilling knowledge from BERT to simple neural networks has been demonstrated recently by Tang et al. (2019). The classification lexicon model could be further improved, e.g. by using phrases or n-grams, and escaping named entities.

In Section 4 a modified version of the classification lexicon model is described that can be used for supervised sentence highlighting in large documents. We would like to investigate the performance of this model in future work.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

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*.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

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)*, volume 1, pages 328–339.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Colm Kearney and Sha Liu. 2014. Textual sentiment in finance: A survey of methods and models. *International Review of Financial Analysis*, 33:171–185.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762.

Quanzhi Li and Sameena Shah. 2017. Learning stock market sentiment lexicon and sentiment-oriented word vector from stocktwits. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 301–310.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance*, 66(1):35–65.

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-volume 1*, pages 142–150. Association for Computational Linguistics.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Nicolas Pröllochs, Stefan Feuerriegel, and Dirk Neumann. 2015. Generating domain-specific dictionaries using bayesian learning. In *ECIS*.

Qiao Qian, Minlie Huang, Jinhao Lei, and Xiaoyan Zhu. 2016. Linguistically regularized lstms for sentiment classification. *arXiv preprint arXiv:1611.03949*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf.*

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

Aliaksei Severyn and Alessandro Moschitti. 2015. On the automatic learning of sentiment lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1397–1402.

Bonggun Shin, Timothy Lee, and Jinho D Choi. 2016. Lexicon integrated cnn models with attention for sentiment analysis. *arXiv preprint arXiv:1610.06272*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Philip J. Stone, Robert F. Bales, J. Zvi Namenwirth, and Daniel M. Ogilvie. 1962. The general inquirer: A computer system for content analysis and retrieval based on the sentence as a unit of information. *Behavioral Science*, 7(4):484–498.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Zhiyang Teng, Duy Tin Vo, and Yue Zhang. 2016. Context-sensitive lexicon features for neural sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1629–1638.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *IJCAI*, pages 1347–1353.

Duy Tin Vo and Yue Zhang. 2016. Don't count, predict! an automatic approach to learning sentiment lexicons for short text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 219–224.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Xin Wang, Yuanchao Liu, SUN Chengjie, Baoxun Wang, and Xiaolong Wang. 2015. Predicting polarities of tweets by composing word embeddings with long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1343–1353.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. *Proc. of HLT-EMNLP-2005*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1046–1056. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

A Hyperparameters

The optimal model hyperparameters, see Table 7, are determined via grid search with evaluation on the respective dev set. The batch size is fixed to 100 and each model is trained until no further dev set accuracy is observed.

Model	**Dataset**	**Hyperparameters**	
Baseline	SST-2	LR	0.05
		d	0.8
		λ	10^{-6}
	FNHL	nested CV	
New	SST-2	LR	0.001
		d	0.7
		I	500
		H	3
	FNHL	nested CV	
NewElmo	SST-2	LR	0.001
		d	0.7
		I	500
		H	3

Table 7: Optimal hyperparameters for each model on the SST-2 dataset. For the FNHL dataset nested cross-validation is used. LR: learning rate for the Adam Optimizer, d: dropout rate, λ: L1 regularization strength, I: number of hidden units, H: number of hidden layers.