Encoding Sentiment Information into Word Vectors for Sentiment Analysis

Zhe Ye♠ **Fang Li**♠ **Timothy Baldwin**♡

♠ Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, 200240, China

♡ School of Computing and Information Systems,
The University of Melbourne, Victoria, 3010, Australia

{yezhejack, fli}@sjtu.edu.cn, tb@ldwin.net

Abstract

General-purpose pre-trained word embeddings have become a mainstay of natural language processing, and more recently, methods have been proposed to encode external knowledge into word embeddings to benefit specific downstream tasks. The goal of this paper is to encode sentiment knowledge into pre-trained word vectors to improve the performance of sentiment analysis. Our proposed method is based on a convolutional neural network and an external sentiment lexicon. Experiments on four popular sentiment analysis datasets show that this method improves the accuracy of sentiment analysis compared to a number of benchmark methods.

1 Introduction

Sentiment analysis plays an important role in many real-world applications. The objective of sentiment classification is to classify a sentence, message or document according to sentiment, often in the form of ordinal regression (e.g. positive vs. neutral vs. negative). In recent years, deep neural networks, such as convolutional neural networks ("CNNs"), have been widely used for sentiment classification. A simple CNN trained over pre-trained word vectors has been shown to achieve highly competitive results (Kim, 2014). Learning task-specific vectors through fine-tuning may offer further gains in performance, and this is the primary focus of this paper.

Separately, there has been recent work on methods for learning word embeddings based not just on textual contexts, but also external knowledge bases (Wieting et al., 2015; Yu and Dredze, 2014; Xu et al., 2014; Bian et al., 2014; Faruqui et al., 2015; Mrkšić et al., 2016; Mrkšić et al., 2017; Vulić et al., 2017). This has also been applied to sentiment classification (Rouvier and Favre, 2016; Yu et al., 2017), with empirical results indicating that explicitly embedding sentiment resources can improve the performance of sentiment analysis.

Existing methods for encoding external knowledge into word vectors are generally trained independently of the downstream task. In order to leverage sentiment lexicons for sentiment analysis, we propose a novel method to combine a feedforward neural network (denoted "SentiNet") with a CNN classifier to encode sentiment knowledge into word vectors during training. The method tunes word vectors through the CNN and SentiNet, based on independent information from supervised training data and sentiment lexicons. Our hypothesis is that joint training of sentiment-targeted word embeddings should improve the overall accuracy of the resulting sentiment analyzer. We conduct several experiments to verify this hypothesis, and compare our method with competitor methods that use antonymy/synonymy lexicons and paraphrase databases.

The major contributions of this paper are as follows: (1) the sentiment lexicon is encoded into word vectors by a feedforward neural network instead of an objective function based on a fixed metric such as cosine similarity or Euclidean distance, and in doing so are able to dynamically learn how to encode the lexicon; (2) word vectors are fine-tuned based on supervised training data and the sentiment lexicon during the training of the CNN sentiment classifier; and (3) we achieve state-of-the-art accuracy over a range of benchmark sentiment analysis datasets.

2 Related Work

2.1 Encoding External Knowledge into Word Vectors

Existing approaches to leveraging external knowledge for word embedding learning for natural language processing fall into two categories: (1) encoding external knowledge during the word vector learning stage; and (2) encoding external knowledge into pre-trained word vectors. Both styles of approach make use of similar linguistic resources such as WordNet (Miller, 1995), FrameNet (Baker et al., 1998), the Paraphrase Database ("PPDB": Ganitkevitch et al. (2013)), or BabelNet (Navigli and Ponzetto, 2012).

Methods in the first category usually change the objective function of the language model or add regularization terms into the original objective function. Yu and Dredze (2014) combine CBOW (Mikolov et al., 2013) with word relations extracted from WordNet and PPDB. Xu et al. (2014) regard relational knowledge and categorical knowledge as learning regularizers, and combine them with the skip-gram objective function. Bian et al. (2014) also combine the objective function of CBOW with external syntactic and semantic knowledge to improve word vectors for extrinsic tasks. These methods all need large unlabeled corpora to learn word vectors from scratch.

In contrast, methods in the second category are lightweight because they adapt pre-trained word vectors via post-processing. This means these methods are compatible with different kinds of word vectors. Wieting et al. (2015) learn PARAGRAM word vectors by fine-tuning over paraphrase data from PPDB. The resultant embeddings outperform the baseline skip-gram embeddings over an extrinsic sentiment analysis task for low-dimensionality word embeddings. Faruqui et al. (2015) use synonym relations extracted from WordNet and other resources to construct an undirected graph. They then retrofit the undirected graph to pre-trained word vectors to obtain new word vectors, under the constraint that the resulting vectors should be close to the vectors of their neighbours in the semantic graph. Antonyms are generally close in vector space, presenting a problem when learning general-purpose word vectors (as in most scenarios, it is undesirable for antonyms to be closely related) (Mnih and Hinton, 2008; Collobert et al., 2011; Mikolov et al., 2013; Levy and Goldberg, 2014; Pennington et al., 2014). In order to solve this problem, antonym lexicons have been used to fine-tune pre-trained word vectors. Mrkšić et al. (2016) present a method called counter-fitting to inject antonymy and synonymy constraints into word vectors trained with GloVe (Pennington et al., 2014) and PARAGRAM (Wieting et al., 2015). The adapted word vectors trained with PARAGRAM achieve the second-highest SimLex-999 (Hill et al., 2015) score. Mrkšić et al. (2017) extend this previous work using negative sampling, to force synonym pairs to be closer to each other than to their negative examples, and forcing antonyms pairs to be further away from each other than from their negative examples.

Encoding external knowledge into word vectors has shown to be effective for improving pre-trained word vectors for intrinsic evaluation such as WordSim-353 (Finkelstein et al., 2002) and SimLex-999. It has also shown to be effective for improving extrinsic tasks such as dialogue state tracking (Mrkšić et al., 2016; Mrkšić et al., 2017; Vulić et al., 2017), sentiment analysis (Faruqui et al., 2015; Wieting et al., 2015; Yu et al., 2017), document classification (Kiela et al., 2015), and word sense disambiguation (Rothe and Schütze, 2015).

The above two kinds of methods both encode external knowledge into word vector space before applying word vectors in downstream tasks. Our method encodes a sentiment lexicon into word vectors when fine-tuning the word vectors in a downstream task.

2.2 Adapting Word Vectors for Sentiment Analysis

There are many methods of adapting word vectors for sentiment analysis. Maas et al. (2011) combine two components — a probabilistic document model and a sentiment component — to jointly learn word vectors. The probabilistic document model does not require labelled data. The sentiment component uses document-level sentiment annotations to constrain words expressing similar sentiment to have similar representations. Tang et al. (2014) changed the objective function of the C&W (Collobert et al., 2011) model to produce sentiment-specific word vectors for Twitter sentiment analysis, by leveraging large volumes of distant-supervised tweets. In order to capture morphological and shape information from words, dos Santos and Gatti (2014) concatenate character-level embeddings and word-level embeddings to form a combined word representation for sentiment analysis. Severyn and Moschitti (2015) refine pre-trained word vectors through a CNN based on distant-supervised data. Zhou et al. (2016) introduced three kinds of word vectors as features into their sentiment classifier. One is general purpose, and the the other two are trained by leveraging sentiment information. However, their feature selection experiments indicate that the task-specific trained word vectors do not improve the performance of their system substantially. Ren et al. (2016) use a recursive autoencoder to learn topic-enhanced word vectors, based on the assumption that the same word can vary in sentiment according to topic. Rouvier and Favre (2016) proposed three kinds of word embeddings — lexical embeddings, part-of-speech embeddings, and sentiment embeddings — in order to train three CNN-based sentiment classifiers. The three classifiers are combined into a fusion model to make the final prediction.

Other methods regard sentiment information as a kind of external knowledge. They encode sentiment information into word vectors by using customized objective functions or introducing regularization terms into objective function of the language model. Yu et al. (2017) utilize a sentiment lexicon to re-rank the nearest neighbors in order to capture sentiment information. The refinement model is based on an objective function which calculates the distance among vectors, in order to adapt word vectors for sentiment analysis. Tang et al. (2016) build on their earlier method (Tang et al., 2014) for Twitter sentiment classification, by leveraging a sentiment lexicon instead of large volumes of distant-supervised Twitter data. Our method also leverages a sentiment lexicon to encode sentiment information into word vectors. The difference between our method and the methods proposed by Yu et al. (2017) and Tang et al. (2014) is that our method applies to word vectors directly via the word embedding layer of the neural network in the context of training a sentiment analyzer, rather than over pre-trained word vectors without explicit task-based training. Our method uses a feedforward neural network to encode sentiment information, as distinct from prior work, which has used cosine similarity or Euclidean distance in the objective function to model word embedding (dis)similarity.

3 Methods

Sentiment lexicons are considered to be a critical component of sentiment analysis. Such external knowledge resources — such as SentiWordNet (Baccianella et al., 2010) and the extended version of Affective Norms of English Words (E-ANEW: Warriner et al. (2013)) — are often used to provide more accurate information about the polarity of a word. Encoding sentiment knowledge into word vectors has been proven to be an effective way to enhance the performance of sentiment analysis.

3.1 Encoding Method

A high-level overview of methods for encoding external knowledge into word vectors for CNN classifiers is presented in Figure 1. Word vectors are used to initialize word embeddings in a CNN classifier. The parameters of the CNN and components for fine-tuning the word embeddings are learned based on supervised training data. The differences in approaches to encoding external knowledge into word vectors are indicated with dotted rectangles in Figure 1. There are three rectangles, representing the two categories of existing approaches described in Section 2.1, and our proposed method: (a) the first class of approach, where word vectors are trained based on external knowledge and unsupervised corpora from scratch; (b) the second class of approach, where pre-trained word vectors are fine-tuned based on external knowledge, and the fine-tuned word vectors are then used to initialize the word embeddings for a CNN classifier; and (c) our method, where we combine external knowledge with pre-trained word vectors during joint parameter training. The parameters of the "Embedding" component are therefore trained not only based on supervised training data, but also based on external knowledge.

Figure 2(a) illustrates the architecture of the CNN classifier proposed by Kim (2014). The input to the Embedding component is a document.[1] The Embedding component of the model outputs a real-valued matrix consisting of the word vectors representing the document. The CNN takes the matrix as input and predicts the sentiment class distribution of the document.

[1]In practice, a sentence in the original paper.

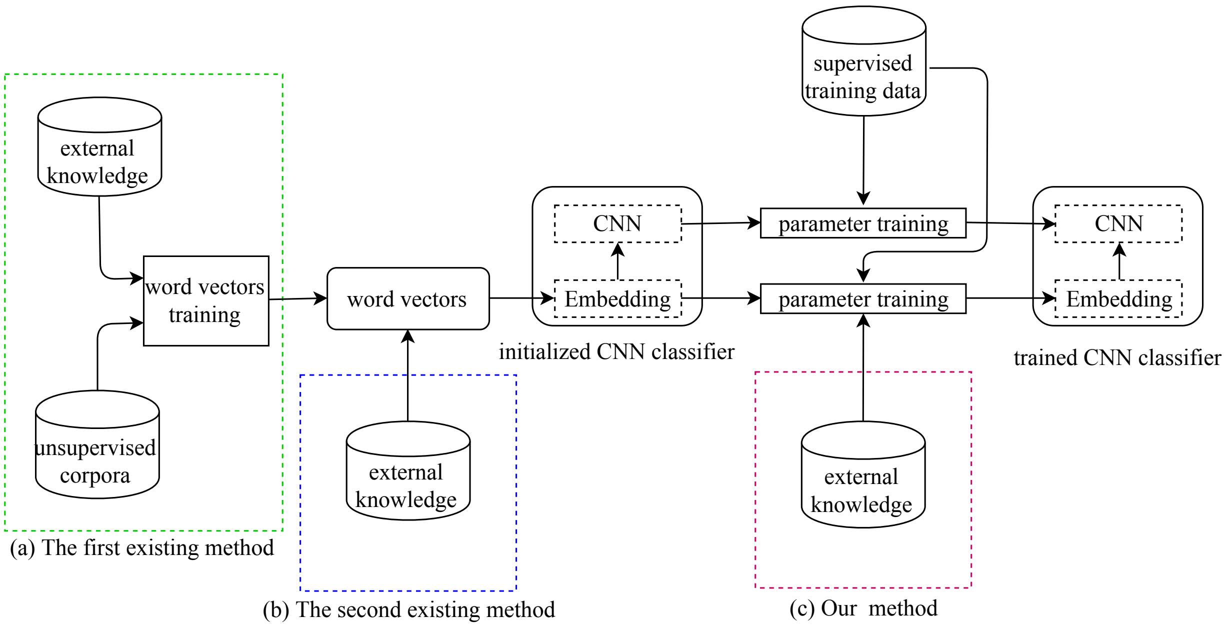
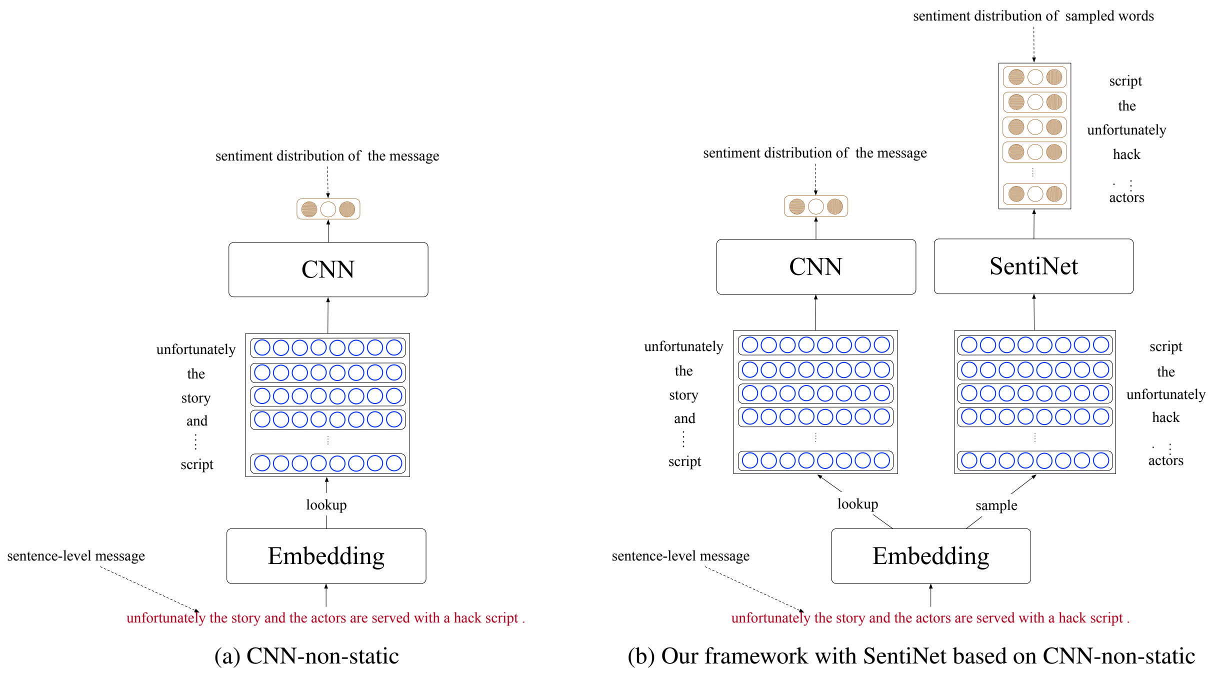


Figure 1: High-level overview of existing methods and our method of encoding external knowledge into word vectors.



sentiment distribution of the message
CNN
unfortunately
the
story
and
script
lookup
sentence-level message
Embedding
unfortunately the story and the actors are served with a hack script .
(a) CNN-non-static
sentiment distribution of sampled words
script
the
unfortunately
hack
actors
sentiment distribution of the message
CNN
SentiNet
unfortunately
the
story
and
script
script
the
unfortunately
hack
actors
lookup
sample
sentence-level message
Embedding
unfortunately the story and the actors are served with a hack script .
(b) Our framework with SentiNet based on CNN-non-static

Figure 2: The framework of CNN-non-static (Kim, 2014) vs. our method for encoding a sentiment lexicon into word vectors by SentiNet.

Figure 2(b) shows the architecture of our method. The Embedding component is initialized with pre-trained word vectors to train the CNN classifier, and a feedforward neural network called "SentiNet" is used to encode sentiment information from SentiWordNet into the word vectors. In Figure 2(b), the input to Embedding is a vector of words contained in the document, denoted by $[w_1, w_2, \ldots, w_N]$. Embedding outputs a real-valued matrix, $\boldsymbol{W} = [\boldsymbol{w}_1; \boldsymbol{w}_2; \ldots; \boldsymbol{w}_N]^\intercal$, which consists of the word vectors to represent the document. The CNN sentiment classifier takes the matrix as input and predicts the sentiment distribution of the document. Let $\boldsymbol{f}^h_m(\boldsymbol{W})$ and $\boldsymbol{f}^g_m(\boldsymbol{W})$ be the prediction and gold-standard distribution of the document. The loss function of the CNN is:

$$\mathcal{L}_{\text{CNN}} = CE(\boldsymbol{f}^h_m(\boldsymbol{W}), \boldsymbol{f}^g_m(\boldsymbol{W}))\,, \tag{1}$$

where $CE(\cdot)$ is a scalar value representing the categorical cross-entropy between the prediction and the gold-standard sentiment distribution. The parameters of the CNN will be updated according to $\mathcal{L}_{\text{CNN}}$. M words are sampled from $[w_1, w_2, \ldots, w_N]$, denoted as $[w_{s_1}, w_{s_2}, \ldots, w_{s_M}]$ where $s_k \in [1, N]$ and $k \in [1, M]$. The word vectors of the sampled words are denoted as $[\boldsymbol{w}_{s_1}, \boldsymbol{w}_{s_2}, \ldots, \boldsymbol{w}_{s_M}]$. SentiNet uses the sampled word vectors to predict the word-level sentiment distribution based on SentiWordNet. Let $\boldsymbol{f}^h_w(\boldsymbol{w}_{s_k})$ and $\boldsymbol{f}^g_w(\boldsymbol{w}_{s_k})$ be the prediction and gold-standard sentiment distribution of the word w_{s_k}. The loss function for SentiNet is:

$$\mathcal{L}_{\text{SentiNet}} = \sum_{k=1}^{M} CE(\boldsymbol{f}^h_w(\boldsymbol{w}_{s_k}), \boldsymbol{f}^g_w(\boldsymbol{w}_{s_k}))\,. \tag{2}$$

The parameters of SentiNet are updated according to $\mathcal{L}_{\text{SentiNet}}$, and the parameters of Embedding are updated according to the combined loss $\mathcal{L}$:

$$\mathcal{L} = \mathcal{L}_{\text{CNN}} + \mathcal{L}_{\text{SentiNet}} \tag{3}$$

Instead of encoding external knowledge into word vectors and training the CNN sentiment classifier separately, the word vectors in our method are fine-tuned with not only the CNN, but also with SentiNet, where SentiNet is regarded as an indicator of word-level sentiment information. If the training of SentiNet converges, the word vectors are considered to have sentiment information.

3.2 SentiNet

Our SentiNet method uses a feedforward neural network with one hidden layer, distinct from previous work which has tended to use an objective function based on cosine similarity or Euclidean distance to capture word embedding similarity. SentiNet takes a word vector $\boldsymbol{w}_{s_k}$ as input, and outputs the sentiment distribution of the word, denoted as $\boldsymbol{f}^h_w(\boldsymbol{w}_{s_k})$. The calculation of the sentiment distribution of the word is based on:

$$\boldsymbol{f}^h_w(\boldsymbol{w}_{s_k}) = \text{softmax}(\boldsymbol{\theta}_2(\sigma(\boldsymbol{\theta}_1 \boldsymbol{w}_{s_k} + \boldsymbol{b}_1)) + \boldsymbol{b}_2)\,, \tag{4}$$

where $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are trainable parameters, and σ is the sigmoid function. The reason for using a feedforward neural network is that it has the same structure as standard word embedding training models such as CBOW, skip-gram and C&W (see Section 2.1).[2]

4 Experiments

4.1 Experimental Setup

Experiments are conducted over four popular, publicly-available sentence-level sentiment classification datasets:

- SemEval2016: The dataset of the SemEval-2016 Message Polarity Classification task. The goal is to classify a given Twitter message according to positive, negative or neutral sentiment (Nakov et al., 2016).

- SemEval2017: The dataset of the SemEval-2017 Message Polarity Classification task. The task formulation is exactly the same as SemEval2016 and the same training data is used, with new test data (Rosenthal et al., 2017).

[2]When training SentiNet with a mini-batch sentences, the words of the mini-batch are classified into positive, neutral and negative according to their sentiment score in the sentiment lexicon. It takes two uniform sample steps to sample a word. First, the word kind (positive, neutral and negative) is sampled uniformly. Then a word is sampled uniformly from the words of the sampled word kind.

	Train			Dev			Test		
	Positive	Neutral	Negative	Positive	Neutral	Negative	Positive	Neutral	Negative
SemEval2016	9169	8098	4115	—	—	—	7059	10342	3231
SemEval2017	19902	22591	7840	—	—	—	2375	5937	3972
MR	5331	—	5331	—	—	—	—	—	—
SST-2	3610	—	3310	444	—	428	909	—	912

Table 1: Statistical breakdown of the four sentiment classification datasets used in this research.

- MR: Single-sentence movie reviews, labeled as having positive or negative sentiment (Pang and Lee, 2005).
- SST-2: A relabelled version of MR, where the overall review is labeled as having positive or negative sentiment (Socher et al., 2013).

Statistics for the four datasets are shown in Table 1. SemEval2016, SemEval2017 and SST-2 have standard training–test splits. MR does not have such a standard split, so we use 10-fold cross validation, consistent with other work on the dataset. We hold out 10% of the training data for SemEval2016, SemEval2017 and MR for development purposes (e.g. for early stopping), whereas SST-2 has a standard development partition. Consistent with standard practice for the respectivce datasets, we evaluate SemEval2016 and SemEval2017 based on macro-averaged F-score ("F1"), and MR and SST-2 based on classification accuracy.

The CNN sentiment classifier is based on the model proposed by Kim (2014). The differences are that: (1) the penultimate layer of our model is not regularized; and (2) we use the Adam optimizer (Kingma and Ba, 2014), whereas Adadelta (Zeiler, 2012) was used in the original. Hyperparameter tuning was performed over the SemEval2016 development data, based on which we use rectified linear units (ReLU), and a dropout rate of 0.5 on the penultimate layer. Three filter window sizes of 3, 4 and 5 are used, each of which contains 100 feature maps. These values are consistent with the hyperparameter settings of Kim (2014).[3]

4.2 Training Variations

We propose three settings for learning the parameters of Embedding, CNN and SentiNet. Learning all parameters together will lead to co-adaption of the parameters, degrading performance. The number of epochs is set to 100. We use the test result of the epoch with highest performance in dev data. The procedure for training the sentiment classifier is divided into two stages, each made up of 50 epochs. There are three variations according to the training time of Embedding and CNN, named BEFORE, DURING and AFTER, which are optionally integrated with the training of SentiNet:

- **BEFORE**: The parameters of Embedding are updated in the first stage (the first 50 epochs), and the parameters of CNN are updated in the second stage (the second 50 epochs).
- **BEFORE+SentiNet**: The parameters of Embedding and SentiNet are updated in the first stage, and the parameters of CNN are updated in the second stage.
- **DURING**: The parameters of Embedding and CNN are updated synchronously across both stages.
- **DURING+SentiNet**: The parameters of Embedding, SentiNet and CNN are updated synchronously across both stages.
- **AFTER**: The parameters of Embedding are updated in the second stage and the parameters of CNN are updated in the first stage.
- **AFTER+SentiNet**: The parameters of Embedding are updated in the second stage. The parameters of CNN and SentiNet are updated in the first stage.

[3]Our code is available at `https://github.com/yezhejack/SentiNet`.

	SemEval2016	SemEval2017	MR	SST-2
Baseline (CNN + Embedding)	0.606	0.640	0.782	0.825
BEFORE	**0.607**	0.640	**0.786**	**0.831**
BEFORE+SentiNet	**0.609**	0.636	**0.788**	**0.834**
DURING	**0.608**	**0.646**	**0.789**	0.825
DURING+SentiNet	**0.610**	0.637	<u>**0.794**</u>	<u>**0.840**</u>
AFTER	**0.614**	**0.650**	**0.787**	**0.834**
AFTER+SentiNet	<u>**0.616**</u>	<u>**0.651**</u>	<u>**0.794**</u>	**0.837**

Table 2: Results of three variations with and without SentiNet. The evaluation metric in the second and third columns is Macro-averaged F-score, while the measurement in the fourth and fifth columns is accuracy. Above-baseline results are indicated in **bold**, and the best result over each dataset is <u>underlined</u>.

4.3 Results

Baseline system: The baseline system consists of Embedding and CNN. The parameters of Embedding are static during training.

Experimental results on the four sentiment classification datasets are reported in Table 2.[4] BEFORE+SentiNet performs better than BEFORE on SemEval2016, MR and SST-2 datasets. DURING+SentiNet performs better than DURING on SemEval2016, MR and SST-2 datasets. AFTER+SentiNet performs better than AFTER over all four datasets. These results show that SentiNet improves the performance of the CNN sentiment classifier. AFTER+SentiNet achieves the best performance on SemEval2016 and SemEval2017, while DURING+SentiNet achieves the best performance on MR and SST-2. AFTER+SentiNet performs better than DURING+SentiNet, as fine-tuning Embedding after CNN and SentiNet can avoid the co-adaption of parameters. BEFORE+SentiNet is the worst because CNN has not been trained when Embedding is tuned, meaning Embedding does not benefit from the task-specific supervised training data.

4.4 Comparison with Other Methods

Experimental setup: We compare our method with three existing methods for encoding external knowledge resources into word embeddings: (1) attract-repel (Mrkšić et al., 2017); (2) PARAGRAM (Wieting et al., 2015); and (3) counter-fitting (Mrkšić et al., 2016). We experiment with word embeddings generated by these three methods to initialize Embedding in our model. These embeddings encode different external knowledges. Embedding and CNN are jointly trained over the supervised training data. For our proposed method, we compare against AFTER+SentiNet and DURING+SentiNet, based on the results from Section 4.3.

The benchmark methods we compare our method against are as follows:

- **word2vec**: The 300 dimensional pre-trained word vectors based on Google News data, and distributed by Google.[5]

- **attract-repel-pos-neg**: The attract-repel[6] method of Mrkšić et al. (2017), which is used to encode the sentiment lexicon into word2vec word embeddings. The sentiment lexicon is the same as the one used in our method.

- **attract-repel-ant-syn**: The attract-repel method of Mrkšić et al. (2017) applied to word2vec word embeddings, based on antonym/synonym lexicons.

[4]Based on McNemar's test, the difference between AFTER-SentiNet and AFTER is significant for SemEval2016 ($p < 0.05$) and SST-2 ($p < 0.01$), but not SemEval2017 or MR ($p > 0.05$).

[5]`https://code.google.com/archive/p/word2vec/`

[6]`https://github.com/nmrksic/attract-repel`

	SemEval2016	SemEval2017	MR	SST-2
word2vec (Mikolov et al., 2013)	0.608	0.647	0.792	0.837
attract-repel-pos-neg (Mrkšić et al., 2017)	0.608	0.641	0.792	0.836
attract-repel-ant-syn (Mrkšić et al., 2017)	0.608	0.626	0.785	0.817
PARAGRAM (Wieting et al., 2015)	0.599	0.632	0.781	**0.839**
counter-fitting (Mrkšić et al., 2016)	0.603	0.627	0.780	**0.838**
DURING+SentiNet	0.601	0.641	0.790	**0.842**
AFTER+SentiNet	**0.613**	**0.648**	**0.796**	**0.844**
SwissCheese (Deriu et al., 2016)	0.633	—	—	—
SENSEI-LIF (Rouvier and Favre, 2016)	0.630	—	—	—
UNIMELB (Xu et al., 2016)	0.617	—	—	—
BB_twtr (Cliche, 2017)	—	0.685	—	—
CNN-non-static (Kim, 2014)	—	0.685	—	—
Re(word2vec) (Yu et al., 2017)	—	—	—	0.879

Table 3: Experiment results of word vectors on the four Sentiment Classification Datasets. The measurement of the second and third columns is Macro F1. The measurement of the fourth and fifth columns is Accuracy.

- **PARAGRAM**: The word vectors of PARAGRAM, generated by encoding PPDB into word2vec word embeddings, based on the pre-trained PARAGRAM word embedding set.[7]
- **counter-fitting**: The word vectors of counter-fitting, generated by encoding an antonym/synonym lexicon into PARAGRAM word vectors, based on the pre-trained word embedding set.[8]

Experimental Analysis: The results of the experiment are shown in Table 3. DURING+SentiNet and AFTER+SentiNet are the results of our method, run on the same sentiment lexicons as other methods.[9]

AFTER+SentiNet performs better than the other methods over all four datasets. The reason is that our word vectors encode not only sentiment information through SentiNet, but also propagate supervision signal from the sentiment analysis task. Comparing the same method with different external knowledge sources, the sentiment lexicon has greater utility than the antonym/synonym lexicon for sentiment classification. Counter-fitting and PARAGRAM outperform word2vec only on SST-2, and not the other three datasets.

The last 6 lines in Table 3 detail state-of-the-art results for the four datasets. SwissCheese (Deriu et al., 2016) leverages large amounts of distant-supervised data to train an ensemble of CNNs. SENSEI-LIF (Rouvier and Favre, 2016) trains three kinds of word embeddings: lexical embeddings, part-of-speech embeddings and sentiment embeddings, in order to train three CNN sentiment classifiers. The three classifiers are combined into a fusion model to make the final prediction. UNIMELB (Xu et al., 2016) consists of a soft-voting ensemble of a language model adapted to classification, a CNN, and a long-short term memory network (LSTM). BB_twtr (Cliche, 2017) also uses a large amount of unlabelled data to pre-train word vectors. It also ensembles CNNs and LSTMs to boost performance. All of these methods are based on ensembling classifiers to improve results, while our method uses a single classifier. Additionally, our CNN model is tuned based on the development data of SemEval2016, and not tuned to the respective datasets. In that sense, the results are highly encouraging, and there is every expectation that ensemble methods would benefit from the incorporation of the outputs of our model.

[7] `https://github.com/nmrksic/counter-fitting/blob/master/word_vectors/paragram.txt.zip`

[8] `https://github.com/nmrksic/counter-fitting/blob/master/word_vectors/counter-fitted-vectors.txt.zip`

[9] In Table 2, the whole vocabulary of pre-trained word2vec is used, whereas in Table 3, we use the intersection of the vocabulary for word2vec, attract-repel-pos-neg, attract-repel-ant-syn, PARAGRAM and counter-fitting. As such, the results are not directly comparable.

5 Conclusion

In this paper, we have proposed a novel method for encoding a sentiment lexicon into word embeddings for sentiment analysis, based on a method we call SentiNet. SentiNet uses a feedforward neural network to lexically encode sentiment information, instead of using an objective function based on cosine similarity or Euclidean distance. Three variations — BEFORE+SentiNet, DURING+SentiNet and AFTER+SentiNet — were proposed to combine SentiNet with Embedding and CNN. Experiments on sentiment classification datasets show that DURING+SentiNet and AFTER+SentiNet are the most effective ways of combining the training of SentiNet with the training of Embedding and the CNN. Our method can combine sentiment information from the training data and the sentiment lexicon to fine-tune word vectors for sentiment classification. For sentiment classification, we have shown sentiment lexicons to have greater utility than antonym/synonym lexicons and paraphrase databases.

In future work, we plan to experiment with encoding other external lexical knowledge beyond sentiment lexicons, and to explore frameworks for encoding external knowledge into deep neural networks.

6 Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61673266).

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2010)*, Valletta, Malta.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL 1998)*, pages 86–90, Montréal, Canada.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Proceedings of the 2014 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2014)*, pages 132–148, Nancy, France.

Mathieu Cliche. 2017. Bb_twtr at SemEval-2017 task 4: Twitter sentiment analysis with CNNs and LSTMs. In *Proceedings of the 11th International Workshop on Semantic Evaluation, (SemEval 2017)*, pages 573–580, Vancouver, Canada.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurélien Lucchi, Valeria De Luca, and Martin Jaggi. 2016. SwissCheese at SemEval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. In *Proceedings of the 10th International Workshop on Semantic Evaluation, (SemEval 2016)*, pages 1124–1128, San Diego, USA.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 69–78, Dublin, Ireland.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2015)*, pages 1606–1615, Denver, USA.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: the paraphrase database. In *Proceedings of Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics (HLT NAACL 2013)*, pages 758–764, Atlanta, USA.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 2044–2048, Lisbon, Portugal.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1746–1751, Doha, Qatar.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), Volume 2: Short Papers*, pages 302–308, Baltimore, USA.

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 (ACL 2011)*, pages 142–150, Portland, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Advances in Neural Information Processing Systems (NIPS 2013)*, pages 3111–3119, Lake Tahoe, USA.

George A. Miller. 1995. WordNet: A lexical database for English. *Commun. ACM*, 38(11):39–41.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems (NIPS 2008)*, pages 1081–1088, Vancouver, Canada.

Nikola Mrkšić, Séaghdha Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2016)*, pages 142–148, San Diego, USA.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gai, Anna Korhonen, and Steve Young. 2017. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the Association for Computational Linguistics*, 5:309–324.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation, (SemEval 2016)*, pages 1–18, San Diego, USA.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 115–124, Ann Arbor, USA.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543, Doha, Qatar.

Yafeng Ren, Ruimin Wang, and Donghong Ji. 2016. A topic-enhanced word embedding for Twitter sentiment classification. *Inf. Sci.*, 369:188–198.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation, (SemEval 2017)*, pages 502–518, Vancouver, Canada.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2015)*, pages 1793–1803, Beijing, China.

Mickael Rouvier and Benoît Favre. 2016. SENSEI-LIF at SemEval-2016 task 4: Polarity embedding fusion for robust sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation, (SemEval 2016)*, pages 202–208, San Diego, USA.

Aliaksei Severyn and Alessandro Moschitti. 2015. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2015)*, pages 959–962, Santiago, Chile.

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, Seattle, USA.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1555–1565.

Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. 2016. Sentiment embeddings with applications to sentiment analysis. *IEEE Trans. Knowl. Data Eng.*, 28(2):496–509.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve J. Young, and Anna Korhonen. 2017. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 56–68, Vancouver, Canada.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 English lemmas. *Behavior Research Methods*, 45(4):1191–1207.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM 2014)*, pages 1219–1228, Shanghai, China.

Steven Xu, Huizhi Liang, and Timothy Baldwin. 2016. UNIMELB at SemEval-2016 tasks 4a and 4b: An ensemble of neural networks and a word2vec based model for sentiment classification. In *Proceedings of the 10th International Workshop on Semantic Evaluation, (SemEval 2016)*, pages 183–189, San Diego, USA.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), Volume 2: Short Papers*, pages 545–550, Baltimore, USA.

Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xue-Jie Zhang. 2017. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 534–539, Copenhagen, Denmark.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. *CoRR*, abs/1212.5701.

Yunxiao Zhou, Zhihua Zhang, and Man Lan. 2016. ECNU at SemEval-2016 task 4: An empirical investigation of traditional NLP features and word embedding features for sentence-level and topic-level sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation, (SemEval 2016)*, pages 256–261, San Diego, USA.