THE JOINT EFFECT OF SEMANTIC AND SYNTACTIC WORD EMBEDDINGS ON SENTIMENT ANALYSIS

Shu Chen, Guang Chen, Wei Wang

Pattern Recognition and Intelligent System Laboratory,
Beijing University of Posts and Telecommunications, Beijing 100876, China
alex.shu.chen@gmail.com, chenguang@bupt.edu.cn, wangwei7175878@bupt.edu.cn

Abstract: Employing pre-trained word embeddings as preliminary features in convolutional neural networks (CNN) for natural language processing (NLP) tasks has been proved to be of benefit. We exploit this idea by taking advantage of different types of word embeddings at the same time. To be specific, we extend CNN models to coordinate two lookup tables, which exploit semantic word embeddings and syntactic word embeddings at the same time. We test our models on several review datasets and all results indicate the positive effect on sentiment analysis. To understand the reason behind, we explore the difference of the two word embeddings and how they influence the CNN models.

Keywords: Word embedding; Semantic and Syntactic similarity; Convolutional neural networks; Sentiment analysis

1 Introduction

Sentiment analysis, also known as opinion mining, is to determine the inherent sentiment of given texts, typically movie or product reviews. Brought out by Bo Pang in 2002 [1], sentiment analysis is generally treated as classification task. However, it turned out to be more challenging. And various methods have been explored [2-4]. Besides traditional binary sentiment classification, it also involves aspect-based sentiment analysis [5], fine-grained sentiment analysis [6], rating prediction [7] and so on.

Deep learning models have won their popularity in computer vision [8] and speech recognition [9]. Although invented for computer vision, CNN has been applied to a wide range of NLP tasks, such as semantic parsing [10], search query retrieval [11], feature mining [12] and traditional tasks like part-of-speech tagging [13]. And various adaptation has been made for CNN to perform better in text classification task. Kim applied a CNN with multiple filter window sizes for sentence classification [14]. Johnson and Zhang incorporated word order information as region features into CNN [15]. Ma et al. modified CNN so as to make use of linguistic structures of sentences rather than surface sequential *n*-grams [16]. Dos Santos and Gatti exploited from character- to sentence-level information to perform sentiment analysis with CNN [17]. Kalchbrenner et al. generalized the max-pooling of CNN to dynamic *k*-max pooling for better sentence modelling [18].

978-1-5090-1246-6/16/$31.00 ©2016 IEEE

As a common practice, the lookup table of CNN models is usually initialized with pre-trained word embeddings. Originally introduced by Bengio et al. [19], word embeddings have exhibited exciting quality as feature extractors. Although their dimension is lower than traditional one-hot representation, they encode far more semantic features of words. And lots of word embedding methods have been proposed [20-23]. Popular as the word embeddings provided by Mikolov et al. are [20], there are still many other word embeddings worth noticing [24-26]. And hence in this paper, we explore more word embeddings and their combination to better the understanding of sentiment.

This paper is organized as follows. In Section 2, we describe two different word embedding methods. In Section 3, we introduce the basic CNN model and extend it to coordinate more word embeddings. In Section 4, we experiment them on four review datasets including movie reviews and product reviews. And In Section 5, we explore the difference of the two word embeddings and how they influence the CNN models, and find a reasonable explanation of it.

2 Word Embedding Methods

We focus on two types of word embeddings in this paper, i.e. semantic word embedding and syntactic word embedding. And the difference between them is demonstrated in Section 5.

2.1 Semantic Word Embedding

The Skip-gram model proposed by Mikolov et al. is an unsupervised neural language model [20]. Trained on large scale text, it maximizes the log-likelihood of the co-occurrence of central words and their linear contexts. Specifically speaking, the training objective is to maximize

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-h\le j\le h,\\ j\neq 0}}\log p(w_{t+j}|w_t) \qquad (1)$$

Where T is the size of training text and h defines the context window size around the central words. From the inner summation, we can see that Skip-gram is trained on sequential contexts. In this way, the produced word embeddings tend to share semantic similarity.

As a popular choice, we downloaded the 300-

dimensional word embeddings trained on Google News Dataset[1]. And we refer to it as **SEM** in this paper.

2.2 Syntactic Word Embedding

Based on the Skip-gram model, Levy and Goldberg proposed a more general idea of arbitrary contexts [24]. The object function can be rewritten as

$$\arg max_{v_w,v_c} \sum_{(w,c)\in D} \log p(D = 1|w, c) \qquad (2)$$

Where *w* is the central word and *c* is its context, *D* is the observed set of (*w*, *c*) pairs in the training text. In this way, it is easier to see that the choice of contexts is actually unrestricted. With a syntactic dependencies parser, the syntactic contexts can be effectively derived. Replaced by syntactic contexts, the model yields more syntactic similarity.

We used the 300-dimensional word embeddings trained on Wikipedia by the authors[2]. We refer to it as **SYN** in this paper.

3 CNN Models

The basic model architecture we used is shown in Figure 1, which is a simple adaptation of Kim's model [14].

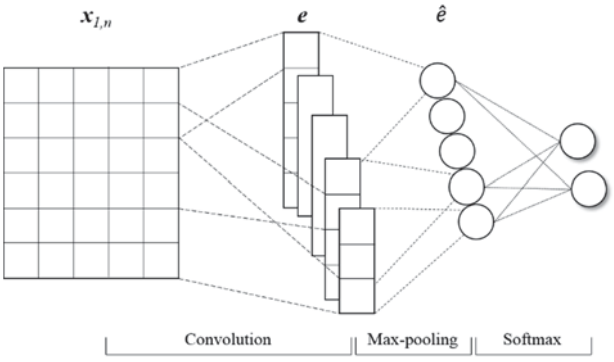

Figure 1 basic CNN model

Suppose the length of an input sentence with necessary paddings is *n*. The one-hot-representation $word_i$ of it is converted to a *k*-dimensional vector $x_i \in \mathbb{R}^k$ by a lookup table operation (not shown in Figure 1). Let $\boldsymbol{x}_{i,j}$ refer to the concatenation of words x_i, x_{i+1}, … , x_{i+j-1}. And $\oplus$ is the concatenation operator.

$$\boldsymbol{x}_{i,j} = x_i \oplus x_{i+1} \oplus \cdots \oplus x_{i+j-1} \qquad (3)$$

A filter $\boldsymbol{W} \in \mathbb{R}^{mk}$ (of window size *m*) is applied to the concatenation of *m* words to produce a new feature *e*, for example,

$$e_i = f\left(\boldsymbol{W} \cdot \boldsymbol{x}_{i,m} + b\right) \qquad (4)$$

Where *f* is a non-linear function such as rectified linear unit and *b* is a bias term. After the convolution of the filter and all possible concatenations, a feature map

$$\boldsymbol{e} = [e_1, e_2, \ldots, e_{n-m+1}] \in \mathbb{R}^{n-m+1} \qquad (5)$$

is generated. And then a max-pooling operation produces $\hat{e} = \max \boldsymbol{e}$ as the feature of the filter. With multiple filter window sizes, different features are extracted and passed to a fully connected softmax layer to make the final classification decision.

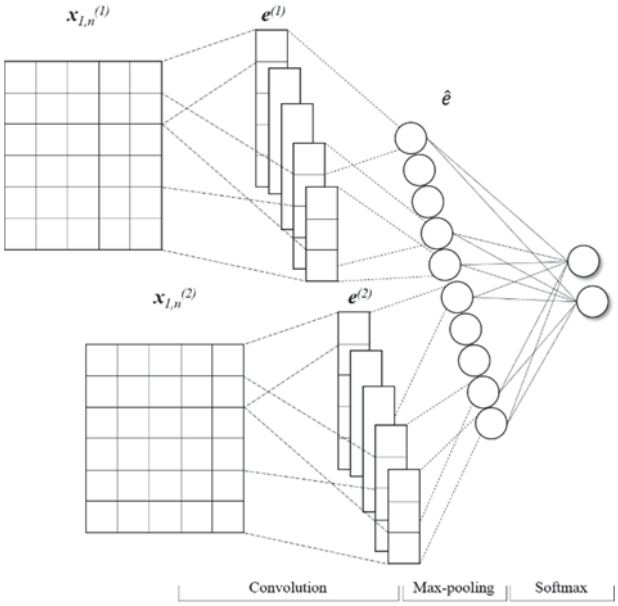

Figure 2 extended CNN model

The lookup table $LT \in \mathbb{R}^{|V|\cdot k}$, where $|V|$ is the vocabulary size of target dataset and *k* is the vector size, is usually initialized with pre-trained word embeddings (or randomly initialized). In order to exploit different word embeddings at the same time, we extended the basic model to coordinate two lookup tables. Since they differ in nature, it might not be a sensible choice to use the same set of filters to convolve with them. And thus we separated operations before softmax (i.e. lookup table, convolution and max-pooling) into two groups, and then concatenated all features produced to form the penultimate layer. The architecture of the extended CNN model is shown in Figure 2.

4 Datasets and Experiments

We tested our models on 4 review datasets for sentiment analysis. Three of them are movie reviews, while the fourth consists of customer reviews of several products.

The **MR** (movie review) dataset contains 10,662 movie reviews, which is created for sentence polarity detection [7]. All reviews are collected from Rotten Tomatoes with their source polarity label (positive or negative). With explicit rating indicators and objective sentences removed, almost all of them are one sentence long. We performed a 2-class polarity classification with this dataset.

The **SST-1** (Stanford Sentiment Treebank) dataset contains 11,855 movie reviews, which is an extension of MR [6]. Every review in MR is parsed by Stanford

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

[2] https://levyomer.wordpress.com/2014/04/25/dependency-based -word-embeddings/

Table I Results of our baseline and extended CNN models

Model	MR		SST-1		SST-2		CR	
	static	non-static	static	non-static	static	non-static	static	non-static
CNN-RAND	/	76.4	/	40.1	/	80.9	/	79.0
CNN-SEM	80.4	81.1	46.6	47.4	85.7	86.4	83.5	83.7
CNN-SYN	78.3	78.8	42.7	44.7	82.8	84.1	79.9	79.9
CNN-SEM&SYN	80.5	**81.3**	48.4	**48.7**	86.1	**86.8**	83.6	**84.4**

* All models are run by GPU.

Parser [27], and some are split into multiple sentences. Fine-grained labels (i.e. very positive, positive, neutral, negative and very negative) are available and the train/dev/test splits are provided. We performed a 5-class polarity classification with this dataset.

The **SST-2** dataset is the same as SST-1 while the neutral reviews are removed and the remaining reviews divided into 2 polarity (positive and negative). And thus there are 9,613 reviews in total. We performed a 2-class polarity classification with this dataset.

The **CR** (customer review) dataset contains 3,757 customer reviews of 14 products (e.g. cameras, MP3s, routers etc.) [5]. The original dataset is intended for aspect-based opinion mining and thus polarity labels are all given by features of products. We only retained reviews if they are labelled and all labels are of the same polarity. We performed a 2-class polarity classification with this dataset.

Since there are no standard train/dev/test splits provided for MR and CR dataset, we performed a 10-fold cross validation to include every review in test set once. 10% of the training data were randomly selected as the dev set, on which we performed early stopping.

Following Kim [14], we used rectified linear unit as the activation function, and multiple filter window sizes (i.e. 3, 4 and 5) with 100 feature maps each. We employed random dropout (whose rate is set to 0.5) on the penultimate layer to regularize the larger than necessary networks. And also, we put an l_2-norms constraint (of 3) on the weight vectors, and selected 50 as the mini-batch size.

For words not included in the vocabulary of pre-trained word embeddings, we randomly generated them by sampling each dimension from $U(-a, a)$, where a is chosen to keep the variance of the vectors the same as the pre-trained ones. And all randomly initialized parameters are kept uniform.

Furthermore, besides keeping pre-trained word embeddings unchanged (**static**) while training models, we also tried fine-tuning them (**non-static**) on the task.

For simplicity, we refer to the randomly initialized CNN model as **CNN-RAND**, the basic CNN model initialized with SEM as **CNN-SEM**, the basic CNN model initialized with SYN as **CNN-SYN** and the extended CNN model initialized with both SEM and SYN as **CNN-SEM&SYN**.

Results of our models are shown in Table I. As we can see, CNN-SEM and CNN-SYN both achieved a marked improvement over the preliminary baseline CNN-RAND. But compared to CNN-SEM, CNN-SYN was not that effective. However, when two word embeddings were combined together (i.e. CNN-SEM&SYN), they yielded the best results. The advantage was most obvious in SST-1. And fine-tuning the word embeddings during training gave further improvements.

5 Further Observations

Due to the difference between SEM and SYN, the performance of models varies. In order to find a rational explanation, we took a close look at 2 interesting parts: the difference between word embeddings and the features selected by models.

5.1 Difference between SEM and SYN

To demonstrate the difference between SEM and SYN, we performed semantic similarity evaluation and syntactic similarity evaluation on them. Instead of using the off-the-shelf word embeddings, we trained them on an English Wikipedia dump[3] containing 1,900 million words, collected in December of 2015. In this way, we eliminated the difference due to the size of training text.

The WordSim353 test collection (**WS**) contains 353 word pairs [28]. The annotators were instructed to assign scores according to the relatedness of each pair. Hence this dataset measures semantic similarity rather than syntactic similarity.

The MEN test collection (**MEN**) contains 3,000 word pairs [29]. And like WS, it is meant for semantic similarity but annotated in a different way.

Different from the above 2 test collections, the SimLex-999 test collection (**SL**) focuses on syntactic similarity instead of semantic similarity [30]. It contains 999 word pairs.

Table II Results of evaluations

		WE1	WE2
Semantic similarity	WS	**0.71**	0.64
	MEN	**0.78**	0.65
Syntactic similarity	SL	0.40	**0.43**

* **WE1** is the embedding trained for SEM,
WE2 is the embedding trained for SYN

We used the cosine distance to measure the similarity between words. And following the routine, we used Spearman to assess the correlation of cosine values and

[3] https://dumps.wikimedia.org/

Table III Cases where one succeeds while the other fails

Movie reviews from SST-2	Major feature selected by CNN-SEM	Major feature selected by CNN-SYN
Cases where CNN-SYN succeeds while CNN-SEM fails		
More a load of enjoyable , Conan-esque claptrap than the punishing , special-effects soul assaults the Mummy pictures represent . (+)	*conan esque claptrap* (-)	*more a load of enjoyable* (+)
It 's the cinematic equivalent of a good page-turner , and even if it 's nonsense , its claws dig surprisingly deep . (+)	*it 's nonsense* (-)	*claws dig surprisingly deep* (+)
Their film falters , however , in its adherence to the Disney philosophy of required poignancy , a salute that I 'd hoped the movie would avoid . (-)	*poignancy , a salute* (+)	*movie would avoid* (-)
The streets , shot by cinematographer Michael Ballhaus , may be as authentic as they are mean , but it is nearly impossible to care about what happens on them . (-)	*cinematographer michael ballhaus* (+)	*is nearly impossible to care* (-)
Cases where CNN-SEM succeeds while CNN-SYN fails		
By presenting an impossible romance in an impossible world , Pumpkin dares us to say why either is impossible -- which forces us to confront what 's possible and what we might do to make it so . (+)	*presenting an impossible romance* (+)	*an impossible romance in* (-)
Provides nail-biting suspense and credible characters without relying on technology-of-the-moment technique or pretentious dialogue . (+)	*provides nail biting suspense* (+)	*technique or pretentious dialogue* (-)
But it pays a price for its intricate intellectual gamesmanship . (-)	*it pays a price for* (-)	*intricate intellectual gamesmanship* (+)
A well acted and well intentioned snoozer . (-)	*snoozer* (-)	*a well acted and* (+)

* The sign in the parentheses indicates whether the review is positive (+) or negative (-).

the gold standards. The results are shown in Table II. As expected, WE1 outrun WE2 in semantic similarity evaluation while WE2 excelled in syntactic similarity evaluation. And now the difference between SEM and SYN was revealed. Trained on sequential contexts, SEM focuses on local information, which leads to topical similarity (i.e. semantic similarity) while the contexts for SYN is directly dependent and thus the word embeddings are more concentrated on functional similarity (i.e. syntactic similarity).

5.2 Features Selected by Models

As the two embeddings exhibit different nature, it is logical to assume that the CNN models initialized with them will display different character as well. To this end, we extracted the features learned by CNNs to be predictive on SST-2.

Both CNN-SEM and CNN-SYN selected a number of features indicating the review polarity, and we extracted these features from the networks. Table III shows some example cases we find interesting. And all these cases were successfully classified by CNN-SEM&SYN. In the first case, CNN-SEM captured the key word *claptrap* but considered it as a negative feature while CNN-SYN managed to locate the positive modifier *enjoyable*. In the penultimate case, however, CNN-SEM made it to discover *pays a price* while CNN-SYN stayed on adjectives *intricate* and *intellectual*.

Based on our massive observation, we noticed that CNN-SYN tends to focus on adjectives and verbs while CNN-SEM has a wider collection of part-of-speech types. It is intuitively simple that adjectives and verbs play an important role in expressing sentiment. However, sentiment analysis is more complicated than that. Solely relying on them does achieve a fairly good result, but the limitation is obvious too. And thus the performance of CNN-SYN is inferior to CNN-SEM. But it is beneficial to use SYN as a supplementary features nonetheless.

6 Conclusions

We extended CNN models to coordinate two lookup tables, which exploit both semantic word embeddings and syntactic word embeddings at the same time. The test results on several sentiment analysis datasets indicate that the combination of the two word embeddings is of noticeable benefit. We further explored how the different word embeddings influence the performance of CNN models on sentiment analysis and found a reasonable explanation. However, there are still some points worth exploring. To combine the effects of different word embeddings, we simply concatenated the produced features together. Some more advanced ensemble methods may bring further improvements. And beyond sentiment analysis, their ability on other tasks such as question classification remains to be discovered.

Acknowledgments

This work was supported by the Natural Science Foundation of China under Grant No.61300080.

References

[1] Pang B, Lee L, Vaithyanathan S. Thumbs up? Sentiment Classification using Machine Learning Techniques. Proceedings of EMNLP, 2002:79-86.

[2] Pang B, Lee L. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. Proceedings of ACL, 2004:271-278.

[3] Yang B, Cardie C. Context-aware Learning for Sentence-level Sentiment Analysis with Posterior Regularization. Proceedings of ACL, 2014:325-335.

[4] Zhou G, Zhao J, Zeng D. Sentiment Classification with Graph Co-Regularization. Proceedings of COLING, 2014:1331-1340.
[5] Hu M, Liu B. Mining and Summarizing Customer Reviews. Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August. 2004:168-177.
[6] Socher R, Perelygin A, Wu J Y, et al. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. Proceedings of EMNLP, 2013:1631-1642.
[7] Pang B, Lee L. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. Proceedings of ACL, 2005:115-124.
[8] Hinton G E, Srivastava N, Krizhevsky A, et al. Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. Computer Science, 2012, 3(4):212-223.
[9] Graves A, Mohamed A, Hinton G. Speech Recognition with Deep Recurrent Neural Networks. Proceedings of ICASSP 2013:6645-6649.
[10] Yih W T, He X, Meek C. Semantic Parsing for Single-Relation Question Answering. Proceedings of ACL, 2014:643-648.
[11] Shen Y, He X, Gao J, et al. Learning Semantic Representations using Convolutional Neural Networks for Web Search. Proceedings of WWW, 2014:373-374.
[12] Xu L H, Liu K, Lai S W, et al. Product Feature Mining: Semantic Clues versus Syntactic Constituents. Proceedings of ACL, 2014:336-346.
[13] Collobert R, Weston J, Bottou L, et al. Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research, 2011, 12(1):2493-2537.
[14] Kim Y. Convolutional Neural Networks for Sentence Classification. Proceedings of EMNLP, 2014:1746-1751.
[15] Johnson R, Zhang T. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. Proceedings of NAACL HLT, 2015:103-112.
[16] Ma M B, Huang L, Xiang B, et al. Dependency-based Convolutional Neural Networks for Sentence Embedding. Proceedings of IJCNLP, 2015:174-179.
[17] Santos C N D, Gattit M. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. Proceedings of COLING, 2014:69-78.
[18] Kalchbrenner N, Grefenstette E, Blunsom P. A Convolutional Neural Network for Modelling Sentences. Proceedings of ACL, 2014:655-665.
[19] Bengio Y, Ducharme R, Vincent P. Neural Probabilistic Language Model. Journal of Machine Learning Research, 2003, 3(6):1137-1155.
[20] Mikolov T, Sutskever I, Chen K, et al. Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems, 2013, 26:3111-3119.
[21] Mnih A, Hinton G E. A Scalable Hierarchical Distributed Language Model. Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December. 2010:1081-1088.
[22] Huang E H, Socher R, Manning C D, et al. Improving word representations via global context and multiple word prototypes. Proceedings of ACL, 2012:873-882.
[23] Pennington J, Socher R, Manning C. Glove: Global Vectors for Word Representation. Proceedings of EMNLP, 2014:1532-1543.
[24] Levy O, Goldberg Y. Dependency-Based Word Embeddings. Proceedings of ACL, 2014:302-308.
[25] Wang L, Dyer C, Black A, et al. Two/Too Simple Adaptations of Word2Vec for Syntax Problems. Proceedings of NAACL HLT, 2015:1299-1304.
[26] Pham N T, Kruszewski G, Lazaridou A, et al. Jointly optimizing word representations for lexical and sentential tasks with the C-PHRASE model. Proceedings of ACL, 2015:971-981.
[27] Klein D, Manning C D. Accurate Unlexicalized Parsing. Proceedings of ACL, 2003:423-430.
[28] Finkelstein L, Gabrilovich E, Matias Y, et al. Placing Search in Context: the Concept Revisited. ACM Transactions on Information Systems, 2002:116-131.
[29] Bruni E, Tran N K, Baroni M. Multimodal Distributional Semantics. Journal of Artificial Intelligence Research, 2014, 49(1):1-47.
[30] Hill F, Reichart R, Korhonen A. SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. Computational Linguistics, 2014, 41(2).