Refining Word Embeddings for Sentiment Analysis

Liang-Chih Yu[1,3], Jin Wang[2,3,4], K. Robert Lai[2,3] and Xuejie Zhang[4]
[1]Department of Information Management, Yuan Ze University, Taiwan
[2]Department of Computer Science & Engineering, Yuan Ze University, Taiwan
[3]Innovation Center for Big Data and Digital Convergence Yuan Ze University, Taiwan
[4]School of Information Science and Engineering, Yunnan University, Yunnan, P.R. China
Contact: lcyu@saturn.yzu.edu.tw

Abstract

Word embeddings that can capture semantic and syntactic information from contexts have been extensively used for various natural language processing tasks. However, existing methods for learning context-based word embeddings typically fail to capture sufficient sentiment information. This may result in words with similar vector representations having an opposite sentiment polarity (e.g., *good* and *bad*), thus degrading sentiment analysis performance. Therefore, this study proposes a word vector refinement model that can be applied to any pre-trained word vectors (e.g., Word2vec and GloVe). The refinement model is based on adjusting the vector representations of words such that they can be closer to both semantically and sentimentally similar words and further away from sentimentally dissimilar words. Experimental results show that the proposed method can improve conventional word embeddings and outperform previously proposed sentiment embeddings for both binary and fine-grained classification on Stanford Sentiment Treebank (SST).

1 Introduction

Word embeddings are a technique to learn continuous low-dimensional vector space representations of words by leveraging the contextual information from large corpora. Examples include C&W (Collobert and Weston, 2008; Collobert et al., 2011), Word2vec (Mikolov et al., 2013a; 2013b) and GloVe (Pennington et al., 2014). In addition to the contextual information, character-level subwords (Bojanowski et al., 2016) and semantic knowledge resources (Faruqui et al., 2015; Kiela et al., 2015) such as WordNet (Miller, 1995) are also useful information for learning word embeddings. These embeddings have been successfully used for various natural language processing tasks.

In general, existing word embeddings are semantically oriented. They can capture semantic and syntactic information from unlabeled data in an unsupervised manner but fail to capture sufficient sentiment information. This makes it difficult to directly apply existing word embeddings to sentiment analysis. Prior studies have reported that words with similar vector representations (similar contexts) may have opposite sentiment polarities, as in the example of *happy-sad* mentioned in (Mohammad et al., 2013) and *good-bad* in (Tang et al., 2016). Composing these word vectors may produce sentence vectors with similar vector representations but opposite sentiment polarities (e.g., a sentence containing *happy* and a sentence containing *sad* may have similar vector representations). Building on such ambiguous vectors will affect sentiment classification performance.

To enhance the performance of distinguishing words with similar vector representations but opposite sentiment polarities, recent studies have suggested learning sentiment embeddings from labeled data in a supervised manner (Maas et al., 2011; Labutov and Lipson, 2013; Lan et al., 2016; Ren et al., 2016; Tang et al., 2016). The common goal of these methods is to capture both semantic/syntactic and sentiment information such that sentimentally similar words have similar vector representations. They typically apply an objective function to optimize word vectors based on the sentiment polarity labels (e.g., positive and negative) given by the training instances. The use of such sentiment embeddings has improved the performance of binary sentiment classification.

This study adopts another strategy to obtain both semantic and sentiment word vectors. Instead of building a new word embedding model from labeled data, we propose a word vector refinement model to refine existing semantically oriented word vectors using sentiment lexicons. That is, the proposed model can be applied to the pre-trained vectors obtained by any word representation learning models (e.g., Word2vec and GloVe) as a post-processing step to adapt the pre-trained vectors to sentiment applications. The refinement model is based on adjusting the pre-trained vector of each affective word in a given sentiment lexicon such that it can be closer to a set of both semantically and sentimentally similar nearest neighbors (i.e., those with the same polarity) and further away from sentimentally dissimilar neighbors (i.e., those with an opposite polarity).

The proposed refinement model is evaluated by examining whether our refined embeddings can improve conventional word embeddings and outperform previously proposed sentiment embeddings. To this end, several deep neural network classifiers that performed well on the Stanford Sentiment Treebank (SST) (Socher et al., 2013) are selected, including convolutional neural networks (CNN) (Kim, 2014), deep averaging network (DAN) (Iyyer et al., 2015) and long-short term memory (LSTM) (Tai et al., 2015; Looks et al., 2017). The conventional word embeddings used in these classifiers are then replaced by our refined versions and previously proposed sentiment embeddings to re-run the classification for performance comparison. The SST is chosen because it can show the effect of using different word embeddings on fine-grained sentiment classification, whereas prior studies only reported binary classification results.

The rest of this paper is organized as follows. Section 2 describes the proposed word vector refinement model. Section 3 presents the evaluation results. Conclusions are drawn in Section 4.

2 Word Vector Refinement

The refinement procedure begins by giving a set of pre-trained word vectors and a sentiment lexicon annotated with real-valued sentiment scores. Our goal is to refine the pre-trained vectors of the affective words in the lexicon such that they can capture both semantic and sentiment information. To accomplish this goal, we first calculate the semantic similarity between each affective word

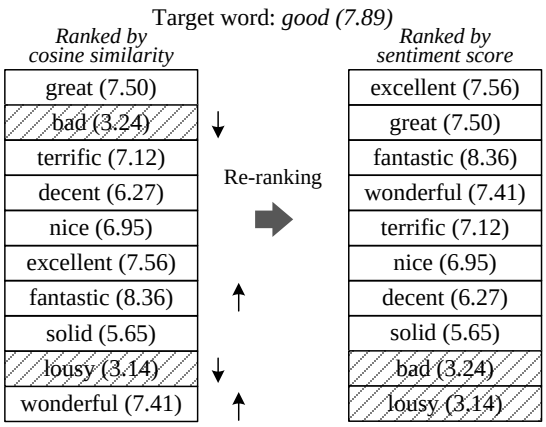

Figure 1: Example of nearest neighbor ranking.

(target word) and the other words in the lexicon based on the cosine distance of their pre-trained vectors, and then select top-*k* most similar words as the nearest neighbors. These semantically similar nearest neighbors are then re-ranked according to their sentiment scores provided by the lexicon such that the sentimentally similar neighbors can be ranked higher and dissimilar neighbors lower. Finally, the pre-trained vector of the target word is refined to be closer to its semantically and sentimentally similar nearest neighbors and further away from sentimentally dissimilar neighbors. The following subsections provide a detailed description of the nearest neighbor ranking and refinement model.

2.1 Nearest Neighbor Ranking

The sentiment lexicon used in this study is the extended version of Affective Norms of English Words (E-ANEW) (Warriner et al., 2013). It contains 13,915 words and each word is associated with a real-valued score in [1, 9] for the dimensions of valence, arousal and dominance. The valence represents the degree of positive and negative sentiment, where values of 1, 5 and 9 respectively denote most negative, neutral and most positive sentiment. In Fig. 1, *good* has a valence score of 7.89, which is greater than 5, and thus can be considered positive. Conversely, *bad* has a valence score of 3.24 and is thus negative. In addition to the E-ANEW, other lexicons such as SentiWordNet (Esuli and Fabrizio, 2006), SoCal (Taboada et al., 2011), SentiStrength (Thelwall et al., 2012), Vader (Hutto et al., 2014), ANTUSD (Wang and Ku, 2016) and SCL-NMA (Kiritchenko and Mohammad, 2016) also provide

real-valued sentiment intensity or strength scores like the valence scores.

For each target word to be refined, the top-*k* semantically similar nearest neighbors are first selected and ranked in descending order of their cosine similarities. In Fig. 1, the left ranked list shows the top 10 nearest neighbors for the target word *good*. The semantically ranked list is then sentimentally re-ranked based on the absolute difference of the valence scores between the target word and the words in the list. A smaller difference indicates that the word is more sentimentally similar to the target word, and thus will be ranked higher. As shown in the right ranked list in Fig. 1, the re-ranking step can rank the sentimentally similar neighbors higher and the dissimilar neighbors lower. In the refinement model, the higher ranked sentimentally similar neighbors will receive a higher weight to refine the pre-trained vector of the target word.

2.2 Refinement Model

Once the word list ranked by both cosine similarity and valence scores for each target word is obtained, its pre-trained vector will be refined to be (1) closer to its sentimentally similar neighbors, (2) further away from its dissimilar neighbors, and (3) not too far away from the original vector.

Let $V = \{v_1, v_2, \ldots, v_n\}$ be a set of the pre-trained vectors corresponding to the affective words in the sentiment lexicon. For each target to be refined, the refinement model iteratively minimizes the distance between the target word and its top-*k* nearest neighbors. The objective function $\Phi(V)$ can thus be defined as

$$\Phi(V) = \sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij}\, dist(v_i, v_j) \qquad (1)$$

where n denotes the total number of vectors in V to be refined, v_i denotes the vector of a target word, v_j denotes the vector of one of its nearest neighbors in the ranked list, $dist(v_i, v_j)$ denotes the distance between v_i and v_j, and w_{ij} denotes the weight of the target word's nearest neighbor, defined as the reciprocal rank of a ranked list. For example, *excellent* in Fig. 1 will receive a weight of 1, *great* will receive a weight of 1/2, and so on. A word ranked higher will receive a higher weight. This weight is used to control the movement direction of the target word towards to its nearest neighbors. That is, the target word will be moved closer to the higher-ranked sentimentally similar neighbors and further away from lower-ranked dissimilar neighbors, as shown in Fig. 2.

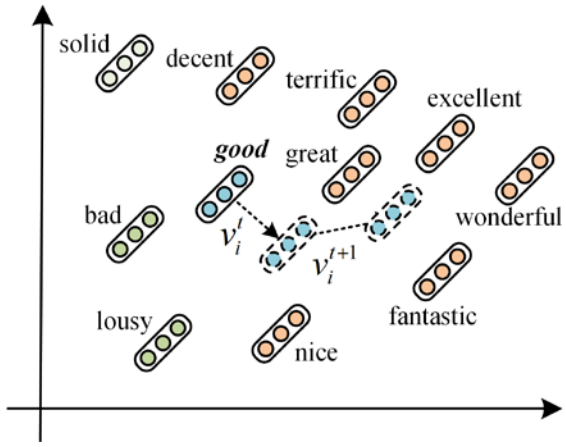

Figure 2: Conceptual diagram of word vector refinement.

To prevent too many words being moved to the same location and thereby producing too many similar vectors, we add a constraint to keep each pre-trained vector within a certain range from its original vector. The objective function is thus divided as two parts:

$$\arg\min \Phi(V) = \arg\min \sum_{i=1}^{n}\left[\alpha\, dist(v_i^{t+1}, v_i^t) + \beta \sum_{j=1}^{k} w_{ij}\, dist(v_i^{t+1}, v_j^t)\right] \qquad (2)$$

where $dist(v_i^{t+1}, v_i^t)$ denotes the distance between the vector of the target word in step t and t+1, i.e., the distance between the refined vector and its original vector. The later one represents the distance between the vector of the target word and that of its neighbors (similar to Eq. (1)). The parameters α and β together are used as a ratio to control how far the refined vector can be moved away from its original vector and toward its nearest neighbors. A greater ratio indicates a stronger constraint on keeping the refined vector closer to its original vector. For the extreme case of α=1 and β=0, the target word will not be moved (refined). As the ratio decreases, the constraint decreases accordingly and the refined vector can be moved closer to its nearest neighbors. The setting of α=0 and β=1 means that the constraint is disabled.

To facilitate the calculation of the partial derivative of $\Phi(V)$, $dist(v_i, v_j)$ in the above equations is measured by the squared Euclidean distance, defined as

$$dist(v_i, v_j) = \sum_{d=1}^{D} (v_i^d - v_j^d)^2 \quad (3)$$

where D is the dimensionality of the word vectors. The global optimal solution of $\Phi(V)$ can be found by using an iterative update method. To do so, we solve the partial derivation of Eq. (2) in step t with respect to word vector v_i^t, and by setting $\frac{\partial \Phi(V)}{\partial v_i^t} = 0$ to obtain a new vector v_i^{t+1} in step t+1. The iterative update procedure is defined as

$$v_i^{t+1} = \frac{\gamma v_i^t + \beta \sum_{j=1}^{k} w_{ij} v_j^t}{\gamma + \beta \sum_{j=1}^{k} w_{ij}} \quad (4)$$

Through the iterative procedure, the vector representation of each target word will be iteratively updated until the change of the location of the target word's vector is converged. The refinement process will be terminated when all target words are refined.

3 Experimental Results

This section evaluates the proposed refinement model, conventional word embeddings and previously proposed sentiment embeddings using several deep neural network models for binary and fine-grained sentiment classification.

Dataset. SST was adopted as the evaluation corpus (Socher et al., 2013). The binary classification subtask (positive and negative) contains 6920/872/1821 samples for the train/dev/test sets, while the fine-grained ordinal classification subtask (very negative, negative, neutral, positive, and very positive) contains 8544/1101/2210 samples of the train/dev/test sets.

Word Embeddings. The word embeddings used for comparison included two conventional word embeddings (GloVe and Word2vec), our refined versions (Re(GloVe) and Re(Word2vec)), and previously proposed sentiment embeddings (HyRank) (Tang et al., 2016). We used the same dimensionality of 300 for all word embeddings.

- GloVe and Word2vec: The respective GloVe and Word2vec (skip-gram) were pre-trained on Common Crawl 840B [1] and GoogleNews[2].
- Re(Glove) and Re(Word2vec): Both the pre-trained GloVe and Word2vec were refined using E-ANEW (Warriner et al., 2013). Each affective word was refined by its top k=10 nearest neighbors with parameters of α:β=0.1 (1:10) (see Eq. (2)).
- HyRank: It was trained using SST, NRC Sentiment140 and IMDB datasets. We compared this method because its code is publicly accessible[3].

Classifiers. The above word embeddings were used by CNN (Kim, 2014) [4], DAN (Iyyer et al., 2015)[5], , bi-directional LSTM (Bi-LSTM) (Tai et al., 2015)[6] and Tree-LSTM (Looks et al., 2017)[7] with default parameter values.

Comparative Results. Table 1 presents the evaluation results of using different word embeddings for different classifiers. For the pre-trained word embeddings, GloVe outperformed Word2vec for DAN, Bi-LSTM and Tree-LSTM, whereas Word2vec yielded better performance for CNN. After the proposed refinement model was applied, both the pre-trained Word2vec and GloVe were improved. The Re(Word2vec) and Re(GloVe) respectively improved Word2vec and GloVe by 1.7% and 1.5% averaged over all classifiers for binary classification, and both 1.6% for fine-grained classification. In addition, both Re(GloVe) and Re(Word2vec) outperformed the sentiment embeddings HyRank for all classifiers on both binary and fine-grained classification, indicating that the real-valued intensity scores used by the proposed refinement model are more effective than the binary polarity labels used by the previously proposed sentiment embedings.

The proposed method yielded better performance because it can remove semantically similar but sentimentally dissimilar nearest neighbors for the target words by refining their vector representations. To demonstrate the effect, we define a measure noise@k to calculate the percentage of top k nearest neighbors with an opposite polarity (i.e., noise) to each word in E-ANEW. For instance, in Fig. 1, the noise@10 for *good* is 20% because there are two words with an opposite polarity to *good* among its top 10 nearest neighbors. Table 2 shows the average noise@10 for different

[1] http://nlp.stanford.edu/projects/glove/
[2] https://code.google.com/archive/p/word2vec/
[3] http://ir.hit.edu.cn/~dytang/
[4] https://github.com/yoonkim/CNN_sentence
[5] https://github.com/miyyer/dan
[6] https://github.com/stanfordnlp/treelstm
[7] https://github.com/tensorflow/fold

Method	Fine-grained	Binary
DAN		
- Word2vec	46.2	84.5
- GloVe	46.9	85.7
- Re(Word2vec)	48.1	87.0
- Re(GloVe)	**48.3**	**87.3**
- HyRank	47.2	86.6
CNN		
- Word2vec	48.0	87.2
- GloVe	46.4	85.7
- Re(Word2vec)	**48.8**	**87.9**
- Re(GloVe)	47.7	87.5
- HyRank	47.3	87.6
Bi-LSTM		
- Word2vec	48.8	86.3
- GloVe	49.1	87.5
- Re(Word2vec)	49.6	88.2
- Re(GloVe)	**49.7**	**88.6**
- HyRank	49.0	87.3
Tree-LSTM		
- Word2vec	48.8	86.7
- GloVe	51.8	89.1
- Re(Word2vec)	50.1	88.3
- Re(GloVe)	**54.0**	**90.3**
- HyRank	49.2	88.2

Table 1: Accuracy of different classifiers with different word embeddings for binary and fine-grained classification.

word embeddings. For the two semantic-oriented word vectors, GloVe and Word2vec, on average around 24% of the top 10 nearest neighbors for each word are noisy words. After refinement, both Re(GloVe) and Re(Word2vec) can reduce noise@10 to around 14%. The HyRank also yielded better performance than both GloVe and Word2vec.

4 Conclusion

This study presents a word vector refinement model that requires no labeled corpus and can be applied to any pre-trained word vectors. The proposed method selects a set of semantically similar nearest neighbors and then ranks the sentimentally similar neighbors higher and dissimilar neighbors lower based on a sentiment lexicon. This ranked list can guide the refinement procedure to iteratively improve the word vector representations.

Word Embeddings	Noise@10 (%)
Word2vec	24.3
GloVe	24.0
HyRank	18.5
Re(Word2vec)	14.4
Re(GloVe)	13.8

Table 2: Average percentages of noisy words in the top 10 nearest neighbors for different word embeddings.

Experiments on SST show that the proposed method yielded better performance than both conventional word embeddings and sentiment embeddings for both binary and fine-grained sentiment classification. In addition, the performances of various deep neural network models have also been improved. Future work will evaluate the proposed method on another datasets. More experiments will also be conducted to provide more in-depth analysis.

Acknowledgments

This work was supported by the Ministry of Science and Technology, Taiwan, ROC, under Grant No. MOST 105-2221-E-155-059-MY2 and MOST 105-2218-E-006-028. The authors would like to thank the anonymous reviewers and the area chairs for their constructive comments.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv: 1607.04606*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML-08)*, pages 160–167.

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

Andrea Esuli, and Fabrizio Sebastiani. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-06)*, pages 417-422.

Manaal Faruqui, Jesse Dodge, Sujay K. 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-15), pages 1606-1615.

Clayton J. Hutto and Eric Gilbert. 2014. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of 8th International AAAI Conference on Weblogs and Social Media*, pages 216-225.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*, pages 1681-1691.

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-15)*, pages 2044-2048.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*, pages 1746-1751.

Svetlana Kiritchenko and Saif M. Mohammad. 2016. The effect of negators, modals, and degree adverbs on sentiment composition. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis at NAACL-HLT 2016*, pages 43-52.

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*, pages 489-493.

Man Lan, Zhihua Zhang, Yue Lu, and Ju Wu. 2016. Three convolutional neural network-based models for learning sentiment word vectors towards sentiment analysis. In *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN-16)*, pages 3172-3179.

Moshe Looks, Marcello Herreshoff, DeLesley Hutchins, and Peter Norvig. 2017. Deep learning with dynamic computation graphs. In *Proceedings of the 5th International Conference on Learning Representations (ICLR-17)*.

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 (ACL-11)*, pages 142-150.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems (NIPS-13)*, pages 1-9.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013b. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR-2013)*, pages 1-12.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39-41.

Saif M. Mohammad, Bonnie J. Dorr, Graeme Hirst, and Peter D. Turney. 2013. Computing lexical contrast. *Computational Linguistics*, 39(3):555-590.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*, pages 1532-1543.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Improving twitter sentiment classification using topic-enriched multi-prototype word embeddings. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 3038-3044.

Richard Socher, Alex Perelygin, and Jy Wu. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*, pages 1631-1642.

Maite Taboada, Julian Brooke, and Kimberly Voll. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*, pages 1556–1566.

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

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *Journal of the Association for Information Science and Technology*, 63(1):163-173.

Shih-Ming Wang and Lun-Wei Ku. 2016. ANTUSD: A large Chinese sentiment dictionary. In *Proc. of the 10th International Conference on Language Resources and Evaluation (LREC-16)*, pages 2697-2702.

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.