Refining Word Embeddings Using Intensity Scores for Sentiment Analysis

Liang-Chih Yu, *Member, IEEE*, Jin Wang, K. Robert Lai, and Xuejie Zhang

***Abstract*—Word embeddings that provide continuous low-dimensional vector representations of words have been extensively used for various natural language processing tasks. However, existing context-based word embeddings such as Word2vec and GloVe typically fail to capture sufficient sentiment information, which may result in words with similar vector representations having an opposite sentiment polarity (e.g., *good* and *bad*), thus degrading sentiment analysis performance. To tackle this problem, recent studies have suggested learning *sentiment embeddings* to incorporate the sentiment polarity (positive and negative) information from labeled corpora. This study adopts another strategy to learn sentiment embeddings. Instead of creating a new word embedding from labeled corpora, we propose a word vector refinement model to refine existing pretrained word vectors using real-valued sentiment intensity scores provided by sentiment lexicons. The idea of the refinement model is to improve each word vector such that it can be closer in the lexicon to both semantically and sentimentally similar words (i.e., those with similar intensity scores) and further away from sentimentally dissimilar words (i.e., those with dissimilar intensity scores). An obvious advantage of the proposed method is that it can be applied to any pretrained word embeddings. In addition, the intensity scores can provide more fine-grained (real-valued) sentiment information than binary polarity labels to guide the refinement process. Experimental results show that the proposed refinement model can improve both conventional word embeddings and previously proposed sentiment embeddings for binary, ternary, and fine-grained sentiment classification on the SemEval and Stanford Sentiment Treebank datasets.**

***Index Terms*—Sentiment analysis, word embeddings, sentiment embeddings, word vector refinement.**

Manuscript received September 12, 2017; revised December 11, 2017; accepted December 11, 2017. Date of publication December 29, 2017; date of current version January 25, 2018. This work was supported by the Ministry of Science and Technology, Taiwan, under Grants MOST 105-2221-E-155-059-MY2 and MOST 105-2218-E-006-028. This paper was presented in part at the Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, Sep. 2017 [1]. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ani Nenkova. *(Corresponding author: Liang-Chih Yu.)*

L. C. Yu is with the Department of Information Management, Yuan Ze University, Taoyuan 32003, Taiwan (e-mail: lcyu@saturn.yzu.edu.tw).

J. Wang is with the Department of Computer Science and Engineering, Yuan Ze University, Taoyuan 32003, Taiwan, and also with the School of Information Science and Engineering, Yunnan University, Kunming 650000, China (e-mail: wangjin@ynu.edu.cn).

K. R. Lai is with the Department of Computer Science and Engineering, Yuan Ze University, Taoyuan 32003, Taiwan (e-mail: krlai@saturn.yzu.edu.tw).

X. Zhang is with the School of Information Science and Engineering, Yunnan University, Kunming 650000, China (e-mail: xjzhang@ynu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2017.2788182

I. Introduction

WORD representation is a fundamental task and has a long history in natural language processing (NLP). Traditionally, a word is represented as a high-dimensional sparse vector of word co-occurrence statistics using distributional approaches [2]–[4] such as vector space models [5]–[8] and the Hyperspace Analog to Language (HAL) model [9]. Recently, a new distributed word representation learning method, known as *word embeddings* [10]–[17] has been developed to represent words as low-dimensional dense vectors of real numbers, which can efficiently capture semantic and syntactic contextual information from very large datasets. These embeddings have been successfully used for various tasks such as named entity recognition [18], word sense disambiguation [19], dependency parsing [20], machine translation [21], text classification [22], and speech processing [23].

Word embeddings learned from a particular context tend to generate similar vector representations for words with similar contexts. This property works well for semantic-oriented applications, but it is problematic for sentiment analysis because words with similar vector representations due to similar contexts may have an opposite sentiment polarity, as in the example of *happy-sad* mentioned in [24] and *good-bad* in [25]. Such examples are not rare. We describe an experiment conducted to calculate the semantic similarity of words in the Affective Norms of English Words (ANEW) [26] lexicon using the cosine measure between their corresponding word vectors obtained by the continuous bag-of-words (CBOW) model [13], [14]. Overall, among the top 10 semantically similar words for each word, on average around 30% of words have an opposite polarity to that of each word. Composing such word vectors may produce ambiguous sentence vectors (e.g., a sentence containing *happy* and a sentence containing *sad* may have similar vector representations), and thus affect sentiment classification performance.

To avoid generating similar vector representations for sentimentally opposite words, recent studies have suggested learning *sentiment embeddings* [25], [27]–[30] to capture both semantic/syntactic and sentiment information such that sentimentally similar words have similar vector representations. Unlike conventional word embeddings learned from unlabeled corpora in an unsupervised manner, the above methods learned sentiment embeddings from labeled corpora in a supervised manner. 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. However, it is difficult

to obtain large manually labeled corpora for some application domains. Although several sentiment embeddings [25], [29], [30] have used a distant supervision method [31], [32] to automatically collect labeled corpora using positive and negative emoticons, such emoticons sometimes are noisy signals.

This study adopts another strategy to learn sentiment embeddings. Instead of creating a new word embedding from labeled corpora, we propose a word vector refinement model to refine existing pre-trained word embeddings using a sentiment lexicon with real-valued intensity or strength scores. The idea of the refinement model is to improve each word vector such that it can be closer in the lexicon to both semantically and sentimentally similar words (i.e., those with similar intensity scores) and further away from sentimentally dissimilar words (i.e., those with dissimilar intensity scores). Compared to existing sentiment embeddings, the proposed method has several advantages. First, it is relatively cost effective to obtain a manually created sentiment lexicon than a manually labeled corpus, while an automatically collected corpus based on emoticons contains a certain amount of noise. Second, the intensity scores can provide more fine-grained (real-valued) sentiment information than binary sentiment polarity labels to guide the refinement process. Third, among most importantly, the proposed refinement model can be applied to any pre-trained word embeddings such as conventional word embeddings (e.g., Word2vec [13], [14] and GloVe [16]) and existing sentiment embeddings. For conventional word embeddings that capture semantic and syntactic information from unlabeled corpora, introducing sentiment information can make them adaptable to sentiment applications. For existing sentiment embeddings that have learned (binary) sentiment polarity information from labeled corpora, the refinement model can provide another type of (fine-grained) sentiment intensity information derived from the sentiment lexicon.

The proposed method is evaluated by examining whether the refinement model can improve conventional word embeddings and existing sentiment embeddings for sentiment classification. Two datasets are used for evaluation, including SemEval-2013 Task 2: Sentiment analysis in Twitter [33] and the Stanford Sentiment Treebank (SST) [34]. The former provides positive, negative and neutral annotations, while the later provides more fine-grained annotations of five classes (very negative, negative, neutral, positive, and very positive). Since prior studies on sentiment embeddings have reported binary and ternary classification results on the SemEval dataset, this study first compares the proposed method on this benchmark dataset and then additionally conducts experiments on the SST dataset to present fine-grained classification results. For classifiers, several deep neural networks with different model architectures are selected, such as the convolutional neural network (CNN) [35] for a convolutional architecture, deep averaging network (DAN) [36] for a multi-layer architecture, long-short term memory (LSTM) [37] for a sequential architecture, and Tree-LSTM [38], [39] for a tree architecture. These models have achieved top performances for the SemEval and SST datasets. Each classifier is implemented using conventional word embeddings, sentiment embeddings and our refined embeddings for performance comparison.

The main contributions of this study are summarized as follows.

1) We propose a word vector refinement model that requires no labeled corpus and can be applied to any pre-trained word embeddings.
2) We use a sentiment intensity lexicon that provides fine-grained (real-valued) sentiment information to improve both conventional word embeddings and sentiment embeddings.
3) We report fine-grained sentiment classification performance on SST for sentiment embeddings.

The rest of this paper is organized as follows. Section II introduces previous work on learning word embeddings and sentiment embeddings. Section III describes the proposed word vector refinement model. Section IV presents the evaluation results. Conclusions are drawn in Section V.

II. Related Work

A. Word Embeddings

Word embeddings focus on learning distributed vector representations of words by leveraging the contextual information in large corpora using neural network architectures. A pioneering work proposed by Bengio *et al.* [10] used a neural network language model (NNLM) to learn word embeddings based on the preceding contexts of each word. Since then, many follow-up studies have been conducted. The C&W model [11], [12] extended the idea of using the preceding contexts alone to incorporate both preceding and succeeding contexts into word embeddings using a convolutional network. Mikolov *et al.* [13], [14] further proposed the CBOW and skip-gram models that use a simple single-layer architecture to enable efficient computation of word embeddings from very large datasets. Levy and Goldberg [15] and Pennington *et al.* [16] respectively indicated that the above methods constructed word embeddings based solely on linear contexts and local contexts (typically a few words in the preceding and succeeding contexts), and thus proposed dependency-based word embeddings and global vectors (GloVe). The dependency-based word embeddings addressed the limitation of linear contexts by introducing syntactic contexts derived from a dependency parser, while the GloVe model addressed the limitation of local contexts by accounting for global word-word co-occurrence statistics. In addition to different types of contextual information, character-level subwords [17] are also useful information for learning word embeddings.

Recent studies have suggested specializing the above general-purpose word embeddings by *retrofitting* the pre-trained word vectors using semantic knowledge resources to enhance downstream semantic applications. Faruqui *et al.* [39] introduced relational knowledge (e.g., synonymy, hypernymy, hyponymy and paraphrase relations) from several semantic lexicons such as WordNet [40], FrameNet [41] and the Paraphrase Database [42] such that semantically linked words were retrofitted to have similar vector representations, thus improving the performance of lexical similarity tasks. This method has been adopted to propose similar medical concepts with a linkage in the Medical Subject Headings (MeSH) ontology having similar vector

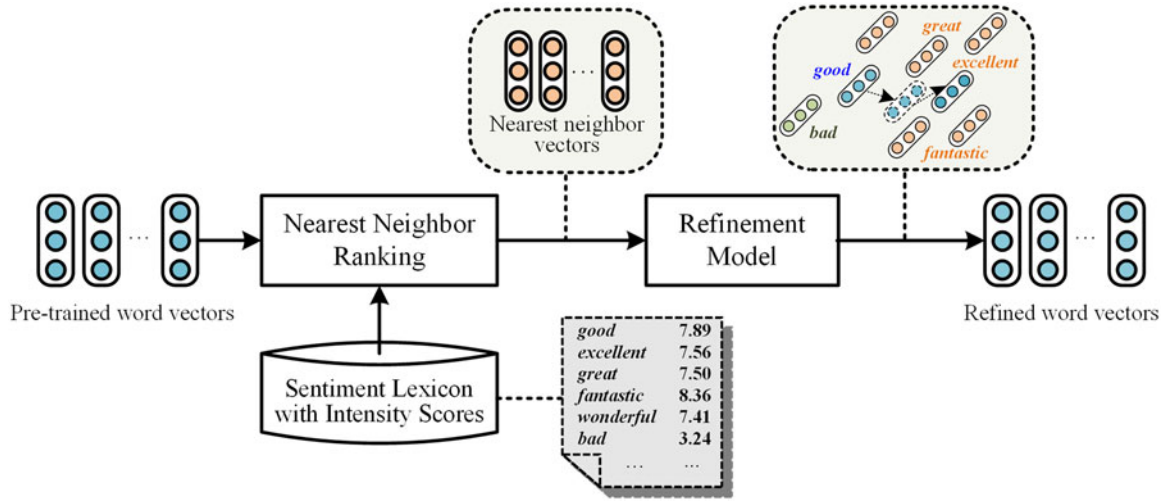


Fig. 1. Overview of the word vector refinement model.

representations [43]. Mrkšić *et al.* [44] proposed a counter-fitting method that injected both antonymy and synonymy relations into vector representations to improve the capability of dialog systems for distinguishing between semantically different but conceptually related words (e.g., *cheaper* and *pricey*). Kiela *et al.* [45] demonstrated that specializing embeddings for similar words (e.g., *dog* and *canine*) was useful for the lexical similarity task, but specializing embeddings for semantically related words (e.g., *dog* and *cat*) could further improve document topic classification performance. Kim *et al.* [46] used semantic intensity information to adjust word embeddings. Bolukbasi *et al.* [47] proposed a de-biasing algorithm to reduce gender bias in word embeddings.

B. Sentiment Embeddings

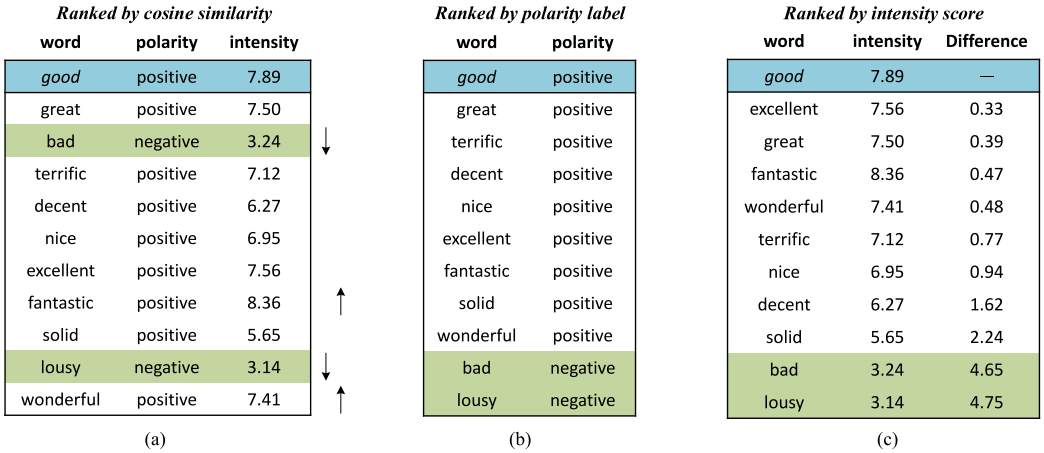
A common characteristic of existing sentiment embeddings is the use of positive and negative polarity labels provided by labeled corpora to guide the learning process through an objective function. Maas *et al.* [27] proposed a semi-supervised method that can capture both semantic information from an unlabeled corpus using a probabilistic document model and sentiment information from a labeled corpus using a predictor function for learning sentiment embeddings. They also introduced a dataset of movie reviews from the Internet Movie Database (IMDB). Labutov and Lipson [28] proposed a fast method to transfer a source embedding to a target embedding by optimizing the conditional likelihood of the polarity labels over the movie review dataset. Recent work has focused on developing neural network models to learn sentiment embeddings from tweets automatically collected by the distant supervision method for Twitter sentiment classification [25], [29], [30]. Tang *et al.* [25] created sentiment embeddings based on the C&W model due to its effectiveness in learning word contexts. They proposed several hybrid models such as the hybrid ranking (HyRank) to incorporate both contexts of words and sentiment polarity of tweets. Ren *et al.* [29] further used C&W and Tang et al's frameworks to learn Multi-prototype Topic and Sentiment-enriched Word Embeddings (M-TSWE). They first incorporated both topic and sentiment information into the single-prototype C&W model and then extended it to multi-prototype. Lan *et al.* [30] extended a conventional CNN architecture with two channels: semantic and sentiment. The semantic and sentiment information captured by these channels were then integrated to create sentiment word vectors (SWV) using three different strategies: mixed (SWV-M), combined (SWV-C) and hybrid (SWV-H).

III. Word Vector Refinement

Given a set of pre-trained word vectors and a sentiment lexicon with intensity scores, the proposed word vector refinement model refines the pre-trained vector of each word in the lexicon. Fig. 1 shows the overall framework of the proposed method, which can be divided into two parts: nearest neighbor ranking and refinement model. For each word (target word) to be refined, the nearest neighbor ranking is first applied to select a set of words both semantically and sentimentally similar to the target word for refinement. This can be accomplished by first calculating the semantic similarity between the target word and the other words in the lexicon based on the cosine distance of their pre-trained vectors. The top-k most similar words are then selected as the nearest neighbors, and ranked according to their intensity scores. The nearest neighbors with a similar intensity score to the target word will be ranked higher and those with a dissimilar intensity score will be ranked lower. Finally, the refinement model will assign different weights to the nearest neighbors according to their rank and use them to adjust the vector representation of the target word to make it closer to its semantically and sentimentally similar nearest neighbors and further away from sentimentally dissimilar neighbors. The following sections provide a detailed description of the nearest neighbor ranking and refinement model.

A. Nearest Neighbor Ranking

This step ranks the semantically similar nearest neighbors by distinguishing their sentimental difference to each target word. Fig. 2 shows the ranking process for the example target word



Ranked by cosine similarity

word	polarity	intensity
good	positive	7.89
great	positive	7.50
bad	negative	3.24
terrific	positive	7.12
decent	positive	6.27
nice	positive	6.95
excellent	positive	7.56
fantastic	positive	8.36
solid	positive	5.65
lousy	negative	3.14
wonderful	positive	7.41

(a)

Ranked by polarity label

word	polarity
good	positive
great	positive
terrific	positive
decent	positive
nice	positive
excellent	positive
fantastic	positive
solid	positive
wonderful	positive
bad	negative
lousy	negative

(b)

Ranked by intensity score

word	intensity	Difference
good	7.89	—
excellent	7.56	0.33
great	7.50	0.39
fantastic	8.36	0.47
wonderful	7.41	0.48
terrific	7.12	0.77
nice	6.95	0.94
decent	6.27	1.62
solid	5.65	2.24
bad	3.24	4.65
lousy	3.14	4.75

(c)

Fig. 2. Example of nearest neighbor ranking. (a) initial ranking (by cosine similarity) (b) polarity-based ranking (c) intensity-based ranking.

good. In this example, the top 10 nearest neighbors of *good* are first selected and ranked in descending order of their cosine similarities, as shown in Fig. 2(a). It can be found that two sentimentally dissimilar neighbors, *bad* and *lousy*, are included in the list. It can also be found that both polarity labels and intensity scores are useful information to distinguish the sentimental difference between the target word and its nearest neighbors. For example, the use of polarity labels can rank the nearest neighbors with the same polarity to the target word higher and those with an opposite lower, as shown in Fig. 2(b). However, the binary polarity information is coarse-grained so that it cannot further distinguish the sentimental difference for words with the same polarity. Therefore, this study uses real-valued intensity scores for nearest neighbor ranking.

The sentiment lexicon used in this study is the extended version of Affective Norms of English Words (E-ANEW) [48]. It contains 13,915 words, each associated with a real-valued score in [1], [9] for the dimensions of valence, arousal and dominance. The valence represents the intensity of positive and negative sentiment, where values of 1, 5 and 9 respectively denote most negative, neutral and most positive sentiment. The valence scores herein are considered as the intensity scores. In Fig. 2, *good* has an intensity score of 7.89, which is greater than 5, and thus can be considered positive. Conversely, *bad* has an intensity score of 3.24 and is thus negative. In addition to the E-ANEW, many other lexicons also provide real-valued sentiment scores for various languages, including SentiWordNet [49], SoCal [50], SentiStrength [51], Vader [52] and SCL-NMA [53] for English, CAWS [54], CVAW [55] and ANTUSD [56] for Chinese, Twitter sentiment lexicon for Arabic [57], and the affective norms for Spanish [58], [59], German [60], [61], French [62], [63], Portuguese [64], Dutch [65], Finnish [66], Italian [67] and Polish [68].

The criterion for intensity-based ranking is the absolute difference of the intensity scores between the target word and its nearest neighbors, as shown in Fig. 2(c). A smaller difference indicates that the neighbor is more sentimentally similar to the target word, and thus will be ranked higher. Conversely, the sentimentally dissimilar neighbors with a quite different intensity score from the target word will be ranked lower. Based on the intensity scores, the sentimental difference among the nearest neighbors can be distinguished even though they have the same polarity.

B. Refinement Model

Once the top-k nearest neighbors for each target word have been obtained, its pre-trained vector will be refined to be 1) closer to the sentimentally similar neighbors, 2) further away from the dissimilar neighbors, and 3) not too far away from the original vector. The refinement model accomplishes these goals by minimizing the distance between each pre-trained word vector and its nearest neighbors based on an objective function. Let $V = \{v_1, v_2, \ldots, v_n\}$ be a set of the pre-trained word vectors, the objective function $\Phi(V)$ is defined as

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

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 top-k nearest neighbors, $\text{dist}(v_i, v_j)$ denotes the distance between the target word's vector v_i and its nearest neighbor's vector v_j, and w_{ij} denotes the weight of a nearest neighbor's vector v_j with respect to the target word's vector v_i.

The $\text{dist}(v_i, v_j)$ is measured by the squared Euclidean distance, defined as

$$\text{dist}(v_i, v_j) = \sum_{d=1}^{D} \left(v_i^d - v_j^d\right)^2 \tag{2}$$

where D is the dimensionality of the word vectors. The w_{ij} is defined as the reciprocal rank of v_j in a ranked list, that is

$$w_{ij} = \frac{1}{\text{rank}_j} \tag{3}$$

where rank_j denotes the rank of v_j generated by the intensity-based ranking process presented in the previous section. Based on this formula, the higher ranked nearest neighbors will receive a higher weight. For example, in Fig. 2(c), with respect to the target word *good*, *excellent* will receive a weight of 1, *great* will

receive a weight of 1/2, and so on. These weights are used to control the moving direction of the target word towards to its nearest neighbors. That is, the target word will be moved toward the higher-ranked sentimentally similar neighbors that have a greater weight and further away from lower-ranked dissimilar neighbors.

To minimize $\Phi(V)$, we use an iterative update method. Equation (1) can thus be re-written as

$$\arg\min \Phi(V) = \operatorname{argmin} \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij} \text{dist}(v_i^{t+1}, v_j^t) \quad (4)$$

where v^t and v^{t+1} respectively denote the vector in step t and $t+1$. To prevent too many words from being moved to the same location and thereby producing too many similar refined vectors, we add a constraint to keep each refined vector within a certain range from its original pre-trained vector. The objective function is thus divided as two parts:

$$\arg\min \Phi(V) = \arg\min \sum_{i=1}^{n} \left[\alpha \text{dist}(v_i^{t+1}, v_i^t) + \beta \sum_{j=1}^{k} w_{ij} \text{dist}(v_i^{t+1}, v_j^t) \right] \quad (5)$$

The first term represents the distance between the vector of the target word in step $t+1$ and t, i.e., the distance between the refined vector (v_i^{t+1}) and its original vector (v_i^t). The second term represents the weighted sum of the distances between the refined vector (v_i^{t+1}) and its nearest neighbors' vectors (v_j^t). The parameters α and β together are used as a ratio (α:β) 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 $\alpha = 1$ and $\beta = 0$, 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. In case of $\alpha = 0$ and $\beta = 1$, the objective functions in (4) and (5) are identical, which means that the constraint is disabled. Fig. 3 illustrates the iterative procedure for the word vector refinement. As shown in the right hand side of Fig. 3, in each iteration the movement direction (the direction of the arrowed line) of the target word is first determined according to the weights (the dashed lines) of its nearest neighbors. The movement distance (the length of the arrowed line) of the target word is then determined based on α and β. For instance, the location of the refined vector in the next iteration will be $v_i^{t+1} = v_i^t$ if $\alpha = 1$ and $\beta = 0$ and $v_i^{t+1} = v_i'^{t+1}$ if $\alpha = 0$ and $\beta = 1$.

The global optimal solution of $\Phi(V)$ can be found by solving the partial derivation of (5) in step t with respect to word vector v_i^t, and set $\frac{\partial \Phi(V)}{\partial v_i^t} = 0$ to obtain a new vector v_i^{t+1} in step $t+1$, that is

$$v_i^{t+1} = \frac{\alpha v_i^t + \beta \sum_{j=1}^{k} w_{ij} v_j^t}{\alpha + \beta \sum_{j=1}^{k} w_{ij}} \quad (6)$$

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 in the lexicon are refined.

Equation (6) shows the update procedure for each single word. To enhance computation efficiency, this procedure is implemented using a matrix notation to update all input word vectors simultaneously. Let $\mathbf{V}^t$ be the matrix of all word vectors in step t, defined as

$$\mathbf{V}^t = \begin{bmatrix} | & & | & & | \\ v_1^t & \cdots & v_j^t & \cdots & v_n^t \\ | & & | & & | \end{bmatrix} \in \mathbb{R}^{d \times n} \quad (7)$$

where each column denotes a target word's vector with a dimensionality of d. Let $\mathbf{W}$ be the matrix of weights, defined as

$$\mathbf{W} = \begin{bmatrix} w_{11} & \cdots & w_{1j} & \cdots & w_{1n} \\ \vdots & & \vdots & & \vdots \\ w_{i1} & \cdots & w_{ij} & \cdots & w_{in} \\ \vdots & & \vdots & & \vdots \\ w_{n1} & \cdots & w_{nj} & \cdots & w_{nn} \end{bmatrix} \in \mathbb{R}^{n \times n} \quad (8)$$

where each row denotes the weights of all neighbors for a target word. It is worth noting that each row has k weights greater than zero because only the top-k nearest neighbors are included in the refinement process.

Based on $\mathbf{V}^t$ and $\mathbf{W}$, the matrix notations of the numerator (denoted as $\mathbf{M}$) and denominator (denoted as $\mathbf{b}$) in (6) can be represented as

$$\mathbf{M} = \alpha \cdot \mathbf{V}^t + \beta \cdot \mathbf{V}^t \mathbf{W}^T$$
$$\mathbf{b} = \alpha + \beta \cdot \mathbf{W}\mathbf{I} \quad (9)$$

where α and β are the above-mentioned control parameters, and $\mathbf{I} = (1, 1, \ldots, 1)^T$ denotes an identity matrix. Each column vector in matrix $\mathbf{M}$ (denoted as $\mathbf{m}_i$) is the numerator term in (6), i.e., $\mathbf{m}_i = \alpha v_i^t + \beta \sum_{j=1}^{k} w_{ij} v_j^t$, and each element in vector $\mathbf{b}$ (denoted as b_i) is the denominator term in (6), i.e., $b_i = \alpha + \beta \sum_{j=1}^{k} w_{ij}$. Given a division function Ψ with $\mathbf{M}$ and $\mathbf{b}$ as inputs, (6) can then be transformed into the following matrix format

$$\mathbf{V}^{t+1} = \Psi(\mathbf{M}^t, \mathbf{b}^t) = \Psi(\alpha \cdot \mathbf{V}^t + \beta \cdot \mathbf{V}^t \mathbf{W}^T, \alpha + \beta \cdot \mathbf{W}\mathbf{I}) \quad (10)$$

where the division function $\Psi(\mathbf{M}, \mathbf{b})$ is implemented by dividing each column vector $\mathbf{m}_i$ in $\mathbf{M}$ by its corresponding element b_i in $\mathbf{b}$. The matrix-based update procedure can converge within a few iterations.

IV. 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, ternary and fine-grained sentiment classification.

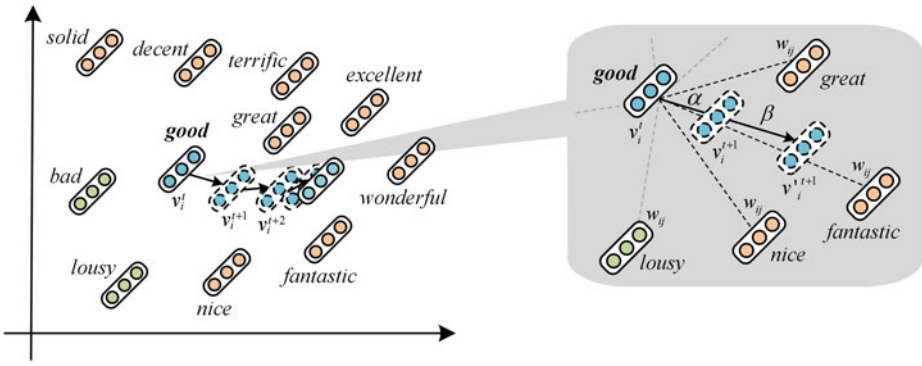


Fig. 3. Conceptual diagram of word vector refinement.

A. Experiment Settings

Datasets: Two datasets, SemEval [33] and SST [34] were used for evaluation. SemEval consists of 9684/1654/3813 tweets for the train/dev/test sets with positive, negative and neutral annotations, while SST consists of 6920/872/1821 sentences with binary annotations and 8544/1101/2210 sentences with fine-grained (five classes) annotations. Since most previously proposed sentiment embeddings used SemEval as a benchmark dataset [25], [29], [30] but none used SST, we first compare the proposed method with the state-of-the-art sentiment embeddings for binary (positive/negative) and ternary (positive/negative/neutral) classification, and then run experiments on the SST to assess performance for binary (positive/negative) and fine-grained (very negative/ negative/neutral/positive/very positive) classification.

Word Embeddings: The word embeddings used for comparison included conventional word embeddings such as Word2vec [14], [15] and GloVe [16], sentiment embeddings such as HyRank [25], M-TSWE [29] and SWV-H [30], and our refined embeddings such as Re(Word2vec), Re(GloVe) and Re(HyRank). Since neither the codes nor the pre-trained word embeddings of M-TSWE and SWV-H are publicly available, these two sentiment embeddings were not run on SST and their refined versions Re(M-TSWE) and Re(SWV-H) were not implemented.

1) *Word2vec and GloVe:* The respective Word2vec (skip-gram)[1] and GloVe[2] and were pre-trained on GoogleNews and Common Crawl 840B with 300 dimensions for both SemEval and SST datasets.
2) *HyRank:* Trained using the sentiment140 corpus [31] with 300 dimensions for SemEval. For SST, we added IMDB and the train/dev sets of SST for training to adapt HyRank for movie review classification. The dimensionality remains 300. The code for training HyRank vectors is publicly accessible.[3]
3) *M-TSWE:* Trained using 10M crawled tweets with 50 dimensions for SemEval.
4) *SWV-H:* Trained using the sentiment140 corpus [31] with 100 dimensions for SemEval.

[1]https://code.google.com/archive/p/word2vec/
[2]http://nlp.stanford.edu/projects/glove/
[3]http://ir.hit.edu.cn/~dytang/

TABLE I
HYPER-PARAMETERS USED IN EACH CLASSIFIER

	SemEval	SST	
	CNN	CNN	DAN, BiLSTM, Tree-LSTM
Filter Number	60	60	–
Filters Length	3	3	–
Pool Length	2	2	–
Hidden Layer Dim.	–	–	120
Optimizer		Adam	
Batch Size		32	
Dropout		0.25	
Epoch		20	

5) *Re(∗):* Denotes the refined embeddings obtained by refining their corresponding pre-trained word vectors using the intensity scores provided by E-ANEW. The dimensionality of the refined vectors is identical to that of their pre-trained vectors. The selection of the optimal parameters settings (k, α and β) is presented in the following section.

Classifiers: For SemEval, we used CNN [35][4] for fair comparison because previously proposed sentiment embeddings also used CNN on this benchmark dataset [29], [30]. For SST, we selected several neural network models with different architectures commonly used for this dataset, including CNN [35], DAN [36][5], Bi-directional LSTM (Bi-LSTM) [37][6] and Tree-LSTM [38].[7] These classifiers were implemented with default parameter settings, as summarized in Table I.

Evaluation Metrics: For SemEval, we adopted the official evaluation metric, macro-averaged F1-score over the positive and negative categories for both binary and ternary Twitter sentiment classification, defined as

$$\text{Macro} - F1 = (F1_{\text{pos}} + F1_{\text{neg}})/2 \tag{11}$$

where $F1_{\text{pos}}$ and $F1_{\text{neg}}$ respectively denote the F1-score for positive and negative categories. The F1-score for each category is calculated as $2 * (precision + recall) / precision * recall$. For

[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

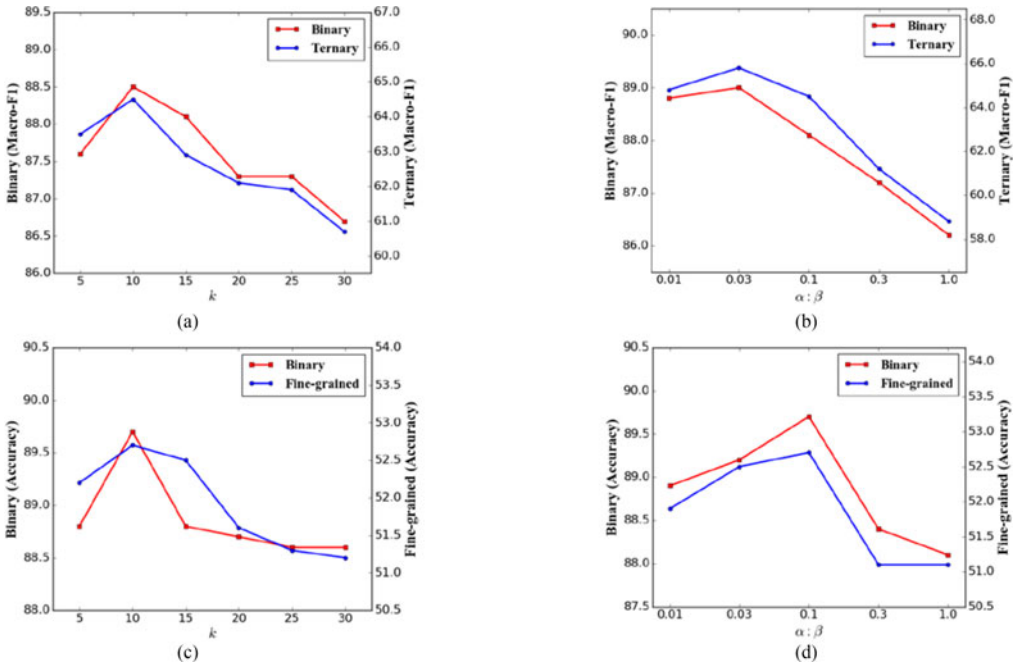


Fig. 4. Parameter selection for the word vector refinement model. (a) Top-k nearest neighbors used in Re(GloVe)$_{\mathrm{CNN}}$ on SemEval (b) Ratio α:β used in Re(GloVe)$_{\mathrm{CNN}}$ on SemEval (c) Top-k nearest neighbors used in Re(GloVe)$_{\mathrm{Tree-LSTM}}$ on SST (d) Ratio α:β used in Re(GloVe)$_{\mathrm{Tree-LSTM}}$ on SST.

SST, the evaluation metric is *accuracy* for both binary and fine-grained sentiment classification.

B. Parameter Selection

The parameters of the proposed method include the number of nearest neighbors k used to guide the movement direction of each pre-trained vector, and the ratio α:β used to control the movement distance of each pre-trained vector from its original location to its nearest neighbors. The optimal settings of these parameters were determined using the development set of SemEval and SST. Fig. 4 shows classification performance against different parameters settings of k and α:β using Re(GloVe)$_{\mathrm{CNN}}$ as an example for SemEval and Re(GloVe)$_{\mathrm{Tree-LSTM}}$ for SST. The results presented in Fig. 4(a) and (c) were obtained by varying k with $\alpha : \beta = 0.1\,(1:10)$, showing that the optimal number of nearest neighbors was $k = 10$ for both SemEval and SST. Once the optimal value is exceeded, performance gradually decreased because including more nearest neighbors may also include more sentimentally dissimilar words to affect the movement direction of the target words. Fig. 4(b) and (d) show the results of varying α:β with $k = 10$. The respective optimal values of α:β were 0.03 (1:30) for SemEval and 0.1 (1:10) for SST, indicating that properly controlling the movement distances of the target words contributed to the refinement process.

Once we have obtained the optimal parameter settings on the development set of SemEval ($k = 10$ and $\alpha : \beta = 0.03$) and SST ($k = 10$ and $\alpha : \beta = 0.1$), they are respectively used for all refined embeddings on the test set of SemEval and SST. Table II shows the performance of Re(GloVe) with the optimal parameter settings on the development set (Fig. 4) and test set of SemEval and SST, showing that their performance on the test set were higher than those on the development set for both SemEval and SST.

TABLE II
RESULTS OF RE(GLOVE) WITH OPTIMAL PARAMETER SETTINGS ON THE DEVELOPMENT SET AND TEST SET OF SEMEVAL AND SST

	SemEval Re(GloVe)$_{\mathrm{CNN}}$		SST Re(GloVe)$_{\mathrm{Tree-LSTM}}$	
	Binary	Ternary	Binary	Fine-grained
Dev. Set (SemEval: 1,654; SST: 1,101)	89.0	65.8	89.7	52.7
Test set (SemEval: 3,813; SST: 2,210)	90.5	68.8	90.3	54.0

TABLE III
MACRO-F1 OF DIFFERENT WORD EMBEDDINGS USED BY CNN FOR BINARY AND TERNARY CLASSIFICATION ON THE SEMEVAL DATASET

Method		Binary	Ternary
Conventional Embeddings	Word2vec	84.6	61.9
	GloVe	86.5	64.4
Sentiment Embeddings	HyRank	86.6	65.3
	M-TSWE	–	66.3
	SWV-H	86.8	–
Refined Embeddings	Re(Word2vec)	88.4	66.5
	Re(GloVe)	90.5	68.8
	Re(HyRank)	89.5	67.3

C. Comparative Results

This section describes three experiments. The first compares the proposed methods to previously published results of the

TABLE IV
ACCURACY OF DIFFERENT WORD EMBEDDINGS USED BY DIFFERENT CLASSIFIERS FOR BINARY AND FINE-GRAINED CLASSIFICATION ON THE SST DATASET

Method		Binary				Fine-grained			
		CNN	DAN	Bi-LSTM	Tree-LSTM	CNN	DAN	Bi-LSTM	Tree-LSTM
Conventional Embeddings	Word2vec	86.8	84.5	86.3	86.7	45.5	46.2	48.8	48.8
	GloVe	85.2	85.7	87.5	89.1	45.1	46.9	49.1	51.8
Sentiment Embeddings	HyRank	85.6	86.6	87.3	88.2	45.8	47.2	49.0	49.2
Refined Embeddings	Re(Word2vec)	**87.6**	87.0	88.2	88.3	**47.7**	48.1	49.6	50.1
	Re(GloVe)	87.1	87.3	**88.6**	**90.3**	47.4	48.3	**49.7**	**54.0**
	Re(HyRank)	86.9	**87.9**	88.1	89.1	47.2	**48.6**	49.3	50.2

TABLE V
RESULTS OF USING POLARITY AND INTENSITY LEXICONS FOR WORD VECTOR REFINEMENT

Sentiment Lexicon	#words	SemEval Re(GloVe)$_{\mathrm{CNN}}$		SST Re(GloVe)$_{\mathrm{Tree-LSTM}}$	
		Binary	Ternary	Binary	Fine-grained
w/o refine	–	86.5	64.4	89.1	51.8
GI$_{\mathrm{Pol}}$	3,457	87.8	65.4	89.4	52.0
EmoLex$_{\mathrm{Pol}}$	5,636	88.3	65.7	89.8	52.6
E-ANEW$_{\mathrm{Pol}}$	13,915	89.5	67.0	90.0	52.8
E-ANEW$_{\mathrm{Int}}$	13,915	90.5	68.8	90.3	54.0

TABLE VI
NOISE@10 FOR DIFFERENT WORD EMBEDDINGS

Method	noise@10 (%)	
Conventional Embeddings	word2vec	24.3
	GloVe	24.0
Sentiment Embeddings	HyRank	18.5
Refined Embeddings	Re(word2vec)	14.4
	Re(GloVe)	13.8
	Re(HyRank)	17.2

state-of-the-art sentiment embeddings for binary and ternary classification on SemEval. The second used SST to examine classification performance, especially fined-grained classification, of different word embeddings for different classifiers. Finally, we investigate the performance difference of using polarity and intensity lexicons for word vector refinement.

Table III shows the results for the SemEval dataset. Both the sentiment embeddings and proposed refined embeddings outperformed the conventional embeddings because both have incorporated sentiment information into vector representations, whereas the conventional embeddings focus on learning contextual information from unlabeled corpora. In addition, the refined embeddings improved both conventional embeddings and sentiment embeddings. For example, Re(Word2vec) and Re(GloVe) respectively improved their pre-trained Word2vec and GloVe by introducing the intensity information from a sentiment lexicon, which makes such conventional context-based word embeddings more adaptable to sentiment analysis. Similarly, Re(HyRank) also improved its pre-trained HyRank using the intensity information even though this sentiment embedding has learned polarity information from labeled corpora before refinement. Another observation is that both Re(Word2vec) and Re(GloVe) yielded better performance than all sentiment embeddings, 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 embeddings. Overall, Re(Word2vec), Re(GloVe) and Re(HyRank) respectively improved Word2vec, GloVe and HyRank by 3.8%, 4.0% and 2.9% for binary classification, and 4.6%, 4.4% and 2.0% for ternary classification.

Table IV presents the results for the SST dataset. Similar to the results on SemEval, the refined embeddings improved both conventional embeddings and sentiment embeddings for all classifiers on both binary and fine-grained classification. Overall, Re(Word2vec), Re(GloVe) and Re(HyRank) respectively improved Word2vec, GloVe and HyRank by 1.7%, 1.5% and 1.1% averaged over all classifiers for binary classification, and 1.6%, 1.6% and 1.0% for fine-grained classification. An inconsistent result between these two datasets is that HyRank did not outperform the conventional embeddings on SST for all classifiers (e.g., Word2vec$_{\mathrm{CNN}}$, GloVe$_{\mathrm{Bi-LSTM}}$ and GloVe$_{\mathrm{Tree-LSTM}}$ for binary classification and GloVe$_{\mathrm{Bi-LSTM}}$ and GloVe$_{\mathrm{Tree-LSTM}}$ for fine-grained classification). One possible reason is that HyRank was trained on a limited size labeled corpora, which is not comparable to the conventional word embeddings trained on very large corpora. Even though the size of the labeled corpora can be increased using a distant supervision method (like SemEval Twitter datasets), such automatically collected corpora may contain noise. This indicates a limitation of directly learning sentiment embeddings from labeled corpora. Conversely, our strategy relies on refining existing word vectors to learn sentiment embeddings, which can be easily used to improve any pre-trained word embeddings.

In addition to intensity scores, the polarity labels can also be used for refinement. As in the example shown in Fig. 2(b), the polarity-based ranking ranks the nearest neighbors with the same polarity to the target word higher and those with an opposite polarity lower. Nearest neighbors with the same polarity is then ranked according to their cosine similarity to the target word. To compare the intensity-based and polarity-based ranking for refinement, we converted the intensity-based E-ANEW lexicon (denoted as E-ANEW$_{\mathrm{Int}}$) into a polarity version (denoted as E-ANEW$_{\mathrm{pol}}$) by assigning a positive label to the words

TABLE VII
TOP-10 NEAREST NEIGHBORS OF SEVERAL EXAMPLE WORDS FOR DIFFERENT WORD EMBEDDINGS

	Top 10 most similar neighbors		
	GloVe	HyRank	Re(GloVe)
satisfied	satisfaction, satisfy, satisfactory, **dissatisfied**, reasonable, **unsatisfied**, pleased, **disappointed**, satisfying, confident	inspiration, praise, productive, tasty, beta, twilight, charming, handsome, virginity, celebrate	satisfaction, satisfy, satisfactory, fulfill, utmost, reasonable, willing, fair, fulfillment, desire
wealthy	millionaire, rich, wealth, aristocratic, billionaire, prosperous, **impoverished**, **greedy**, privileged, businessman	top, lunch, become, pool, swim, familiar, fan, word, learn, official	wealth, prosperous, riches, prosperity, fortune, rich, privileged, privilege, thriving, treasure
strong	strength, **weak**, good, robust, solid, **tough**, consistent, powerful, confident, tremendous	frustrating, depressed, unfair, battery, sunburn, wreck, bummer, cloudy, surgery, error	robust, solid, strength, consistent, stable, efficient, decent, capable, responsive, fair
incomplete	**pass**, incorrect, inaccurate, inconsistent, **complete**, rush, questionable, **partial**, erroneous, inadequate	**beard**, toilet, beg, insist, rear, embarrassment, warehouse, mow, illogical, dislike	incorrect, erroneous, inaccurate, inconsistent, **correct**, inadequate, faulty, doubtful, improper, insufficient
unlucky	unfortunate, **lucky**, miserable, misfortune, hapless, unhappy, **fortunate**, **luck**, poor, hopeless	**replacement**, **camper**, ashamed, **laughable**, lousy, lump, miserable, crushed, poop, freezing	sad, unfortunate, miserable, disappointing, shame, afraid, bad, scared, dire, hopeless

with an intensity score greater than or equal to 5 and a negative label to those with an intensity score below 5. Several popular polarity lexicons such as General Inquirer (GI) [69] and NRC Word-Emotion Association Lexicon (EmoLex) [70], [71] were also included to examine refinement performance for different sentiment lexicons. Table V presents the results. The respective Re(GloVe)$_{\mathrm{CNN}}$ and Re(GloVe)$_{\mathrm{Tree-LSTM}}$ were taken as an example for SemEval and SST. The row labeled "w/o refine" represents the results without refinement, i.e., the results of GloVe$_{\mathrm{CNN}}$ and GloVe$_{\mathrm{Tree-LSTM}}$. The column labeled "#words" denotes the number of positive and negative words (excluding neutral words) in the lexicons used for refinement. Refinement performance increased with the number of affective words used for refinement. Once E-ANEW$_{\mathrm{Int}}$ was used, the performance was further improved, especially for fine-grained classification because the real-valued intensity scores can provide more fine-grained sentiment information than binary polarity information for distinguishing sentimental difference between words.

D. Discussion

The refined embeddings yielded better performance because they 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. 2, the noise@10 for *good* is 20% because there are two noisy words with an opposite polarity to *good* among its top 10 nearest neighbors. Table VI shows the average noise@10 for different word embeddings. The noise@10 of the respective conventional embeddings and sentiment embeddings were around 24% and 18.5%. After refinement, the noisy words were removed and their noise@10 was respectively reduced to around 14% and 17%. Both Re(Word2vec) and Re(GloVe) yielded greater reduction of noise@10 than Re(HyRank). This is the reason why the classification performance improvement for both Re(Word2vec) over Word2vec and of Re(GloVe) over GloVe) was greater than that of Re(HyRank) over HyRank.

The refined embeddings can remove the noisy words because they can refine the pre-trained word vectors closer to their sentimentally similar nearest neighbors and further away from sentimentally dissimilar neighbors. To explain the refinement effect from a qualitative viewpoint, Table VII presents the 10 nearest neighbors of several example words for different word embeddings, showing that the top-ranked noisy words (marked in bold) suggested by GloVe have been successfully removed by Re(GloVe). While HyRank learns new embeddings from labeled corpora and thus may produce quite different nearest neighbors, Re(GloVe), based on the pre-trained GloVe, retains more semantically and sentimentally similar nearest neighbors from GloVe.

V. 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 uses a sentiment intensity lexicon that can provide real-valued sentiment information to rank a set of both semantically and sentimentally similar nearest neighbors for each word. The ranked nearest neighbors are then used to guide the movement direction and distance of the refinement procedure to iteratively improve each word vector representation. Experiments on SemEval and SST show that the proposed method can improve both conventional word embeddings and sentiment embeddings for binary, ternary and fine-grained sentiment classification. In addition, the performance of various deep neural network models was also improved. Future work will extend the proposed method using multiple sentiment lexicons and more effective ranking methods to improve sentiment classification performance.

REFERENCES

[1] L. C. Yu, J. Wang, K. R. Lai, and X. Zhang, "Refining word embeddings for sentiment analysis," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 545–550.
[2] Z. Harris, "Distributional structure," *Word*, vol. 10, no. 2–3, pp. 146–62, 1954.

[3] G. A. Miller and W. G. Charles, "Contextual correlates of semantic similarity," *Lang. Cognitive Processes*, vol. 6, no. 1, pp. 1–28, 1991.

[4] M. Baroni, "Composition in distributional semantics," *Lang. Linguistics Compass*, vol. 7, no. 10, pp. 511–522, 2013.

[5] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *J. Artif. Intell. Res.*, vol. 37, no. 1, pp. 141–188, 2010.

[6] J. Mitchell and M. Lapata, "Composition in distributional models of semantics," *Cognitive Sci.*, vol. 34, pp. 1388–1429, 2010.

[7] K. Erk, "Vector space models of word meaning and phrase meaning: A survey," *Lang. Linguistics Compass*, vol. 6, no. 10, pp. 635–653, 2012.

[8] S. Clark, "Vector space models of lexical meaning," in *Handbook of Contemporary Semantics*, 2nd ed., S. Lappin and C. Fox, Eds. Malden, MA, USA: Wiley, 2015, pp. 493–522.

[9] C. Burgess, K. Livesay, and K. Lund, "Explorations in context space: Words, sentences, discourse," *Discourse Processes*, vol. 25, no. 2/3, pp. 211–257, 1998.

[10] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

[11] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 160–167.

[12] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. Int. Conf. Learn. Representations*, 2013, pp. 1–12.

[14] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[15] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *Proc. ACL*, 2014, pp. 302–308.

[16] J. Pennington, R. Socher, and C. D Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1532–1543.

[17] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," arXiv: 1607.04606.

[18] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: A simple and general method for semi-supervised learning," in *Proc. 48th Annu. Meeting Assoc. Comput. Linguistics*, 2010, pp. 384–394.

[19] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multiple word prototypes," in *Proc. 50th Annu. Meeting Assoc. Comput. Linguistics, Long Papers*, 2012, pp. 873–882.

[20] M. Bansal, K. Gimpel, and K. Livescu, "Tailoring continuous word representations for dependency parsing," in *Proc. ACL*, 2014, pp. 809–815.

[21] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Empirical Methods Natural Lang. Process.*, 2014, pp. 1724–1734.

[22] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," arXiv:1607.01759.

[23] G. Mesnil *et al.*, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 3, pp. 530–539, Mar. 2015.

[24] S. M. Mohammad, B. J. Dorr, G. Hirst, and P. D. Turney, "Computing lexical contrast," *Comput. Linguistics*, vol. 39, no. 3, pp. 555–590, 2013.

[25] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, and M. Zhou, "Sentiment embeddings with applications to sentiment analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 2, pp. 496–509, Feb. 2016.

[26] M. M. Bradley and P. J. Lang, "Affective norms for English words (ANEW): Instruction manual and affective ratings," The Center for Research in Psychophysiology, Univ. of Florida, Gainesville, FL, USA: Tech. Rep. C-1, 1999.

[27] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. ACL*, 2011, pp. 142–150.

[28] I. Labutov and H. Lipson, "Re-embedding words," in *Proc. ACL*, 2013, pp. 489–493.

[29] Y. Ren, Y. Zhang, M. Zhang, and D. Ji, "Improving Twitter sentiment classification using topic-enriched multi-prototype word embeddings," in *Proc. AAAI*, 2016, pp. 3038–3044.

[30] M. Lan, Z. Zhang, Y. Lu, and J. Wu, "Three convolutional neural network-based models for learning sentiment word vectors towards sentiment analysis," in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 3172–3179.

[31] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," CS224N Project Rep., Computer Science, Stanford, CA, USA, 2009, pp. 1–12.

[32] M. Purver and S. Battersby, "Experimenting with distant supervision for emotion classification," in *Proc. Eur. Ch. Assoc. Comput. Linguistics*, 2012, pp. 482–491.

[33] P. Nakov, S. Rosenthal, Z. Kozareva, V. Stoyanov, A. Ritter, and T. Wilson, "Semeval-2013 task 2: Sentiment analysis in Twitter," in *Proc. SemEval*, 2013, pp. 312–320.

[34] R. Socher *et al.*, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2013, pp. 1631–1642.

[35] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1746–1751.

[36] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daum´e III1, "Deep unordered composition rivals syntactic methods for text classification," in *Proc. ACL-IJCNLP*, 2015, pp. 1681–1691.

[37] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proc. ACL*, 2015, pp. 1556–1566.

[38] M. Looks, M. Herreshoff, D. Hutchins, and P. Norvig, "Deep learning with dynamic computation graphs," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–12.

[39] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, Retrofitting word vectors to semantic lexicons, in *Proc. Annu. Conf. North Amer. Ch. Assoc. Comput. Linguistics*, 2015, pp. 1606–1615.

[40] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[41] C. F. Baker, C. J. Fillmore, and J. B. Lowe, "The Berkeley FrameNet Project," in *Proc. ACL*, 1998, pp. 86–90.

[42] J. Ganitkevitch, B. Van Durme, and C. Callison-Burch, "PPDB: The paraphrase database," in *Proc. Annu. Conf. North Amer. Ch. Assoc. Comput. Linguistics*, 2013, pp. 758–764.

[43] Z. Yu, T. Cohen, E. V. Bernstam, and B. C. Wallace, "Retrofitting word vectors of mesh terms to improve semantic similarity measures," in *Proc. 7th Int. Workshop Health Text Mining Inf. Anal.*, 2016, pp. 43–51.

[44] N. Mrkšić *et al.*, "Counter-fitting word vectors to linguistic constraints," in *Proc. Annu. Conf. North Amer. Ch. Assoc. Comput. Linguistics*, 2016, pp. 142–148.

[45] D. Kiela, F. Hill, and S. Clark, "Specializing word embeddings for similarity or relatedness," in *Proc. Empirical Methods Natural Lang. Process.*, 2015, pp. 2044–2048.

[46] J. K. Kim, M. C. de Marneffe, and E. Fosler-Lussier, "Adjusting word embeddings with semantic intensity orders," in *Proc. 1st Workshop Representation Learn. NLP*, 2016, pp. 62–69.

[47] T. Bolukbasi, K. W. Chang, J. Zou, V. Saligrama, and A. Kalai, "Man is to computer programmer as woman is to homemaker? Debiasing word embeddings," in *Proc. Neural Inf. Process. Syst.*, 2016, pp. 4349–4357.

[48] A. B. Warriner, V. Kuperman, and M. Brysbaert, "Norms of valence, arousal, and dominance for 13,915 English lemmas," *Behav. Res. Methods*, vol. 45, no. 4, pp. 1191–1207, 2013.

[49] A. Esuli and F. Sebastiani, "SentiWordNet: A publicly available lexical resource for opinion mining," in *Proc. LREC*, 2006, pp. 417–422.

[50] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Comput. Linguistics*, vol. 37, no. 2, pp. 267–307, 2011.

[51] M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment strength detection for the social web," *J. Assoc. Inf. Sci. Technol.*, vol. 63, no. 1, pp. 163–173, 2012.

[52] C. J. Hutto and E. Gilbert, "VADER: A parsimonious rule-based model for sentiment analysis of social media text," in *Proc. 8th Int. AAAI Conf. Weblogs Social Media*, 2014, pp. 216–225.

[53] S. Kiritchenko and S. M. Mohammad, "The effect of negators, modals, and degree adverbs on sentiment composition," in *Proc. WASSA*, 2016, pp. 43–52.

[54] Y. N. Wang, L. M. Zhou, and Y. J. Luo, "The pilot establishment and evaluation of Chinese affective word system," *Chin. Mental Health J.*, vol. 22, no. 8, pp. 608–612, 2008.

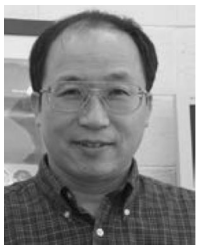
[55] L. C. Yu *et al.*, "Building Chinese affective resources in valence-arousal dimensions," in *Proc. NAACL/HLT*, 2016, pp. 540–545.

[56] S. M. Wang and L. W. Ku, "ANTUSD: A large Chinese sentiment dictionary," in *Proc. LREC*, 2016, pp. 2697–2702.

[57] S. Kiritchenko, S. M. Mohammad, and M. Salameh, "SemEval-2016 Task 7: Determining sentiment intensity of English and Arabic phrases," in *Proc. SemEval*, 2016, pp. 42–51.
[58] J. Redondo, I. Fraga, I. Padrón, and M. Comesaña, "The Spanish adaptation of ANEW (Affective Norms for English Words)," *Behav. Res. Methods*, vol. 39, no. 3, pp. 600–605, 2007.
[59] M. Guasch, P. Ferré, and I. Fraga, "Spanish norms for affective and lexico-semantic variables for 1,400 words," *Behav. Res. Methods*, vol. 48, no. 4, pp. 1358–1369, 2016.
[60] M. L. H. Võ *et al.*, "The Berlin Affective Word List Reloaded (BAWL-R)," *Behav. Res. Methods*, vol. 41, no. 2, pp. 534–538, 2009.
[61] D. S. Schmidtke, T. Schröder, A. M. Jacobs, and M. Conrad, "ANGST: Affective norms for German sentiment terms, derived from the affective norms for English words," *Behav. Res. Methods*, vol. 46, no. 4, pp. 1108–1118, 2014.
[62] A. L. Gilet, D. Grühn, J. Studer, and G. Labouvie-Vief, "Valence, arousal, and imagery ratings for 835 French attributes by young, middle-aged, and older adults: The French Emotional Evaluation List (FEEL)," *Eur. Rev. Appl. Psychol.*, vol. 62, no. 3, pp. 173–181, 2012.
[63] C. Monnier and A. Syssau, "Affective norms for French words (FAN)," *Behav. Res. Methods*, vol. 46, no. 4, pp. 1128–1137, 2014.
[64] A. P. Soares, M. Comesaña, A. P. Pinheiro, A. Simões, and C. S. Frade, "The adaptation of the Affective Norms for English Words (ANEW) for European Portuguese," *Behav. Res. Methods*, vol. 44, no. 1, pp. 256–269, 2012.
[65] A. Moors *et al.*, "Norms of valence, arousal, dominance, and age of acquisition for 4,300 Dutch words," *Behav. Res. Methods*, vol. 45, no. 1, pp. 169–177, 2013.
[66] C. Söderholm, E. Häyry, M. Laine, and M. Karrasch, "Valence and Arousal ratings for 420 Finnish nouns by age and gender," *PLOS ONE*, vol. 8, no. 8, 2013, Art. no. e72859.
[67] M. Montefinese, E. Ambrosini, B. Fairfield, and N. Mammarella, "Semantic memory: A feature-based analysis and new norms for Italian," *Behav. Res. Methods*, vol. 45, no. 2, pp. 440–461, 2013.
[68] M. Riegel *et al.*, "Affective Word List (NAWL): The cultural adaptation of the Berlin Affective Word List–Reloaded (BAWL-R) for Polish," *Behav. Res. Methods*, vol. 47, no. 4, pp. 1222–1236, 2015.
[69] P. J. Stone and E. B. Hunt, "A computer approach to content analysis: Studies using the general inquirer system," in *Proc. May 21–23, 1963, Spring Joint Comput. Conf.*, 1963, pp. 241–256.
[70] S. M. Mohammad and P. D. Turney, "Crowdsourcing a word-emotion association lexicon," *Computat. Intell.*, vol. 29, no. 3, pp. 436–465, 2013.
[71] S. M. Mohammad and P. D. Turney, "Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon," in *Proc. NAACL-HLT 2010 Workshop Comput. Approaches Anal. Gener. Emotion Text*, 2010, pp. 26–34.

Liang-Chih Yu received the Ph.D. degree in computer science and information engineering from the National Cheng Kung University, Tainan, Taiwan. He is an Associate Professor in the Department of Information Management, Yuan Ze University, Taoyuan, Taiwan. From 2007 to 2008, he was a Visiting Scholar at the Natural Language Group, Information Sciences Institute, University of Southern California (USC/ISI). He is currently a Board Member of the Association for Computational Linguistics and Chinese Language Processing (ACLCLP), and serves as an editorial board member of the *International Journal of Computational Linguistics and Chinese Language Processing*. His research interests include natural language processing, sentiment analysis, information retrieval, and text mining.

Jin Wang received the Ph.D. degree in computer science and engineering from Yuan Ze University, Taoyuan, Taiwan and in communication and information systems from Yunnan University, Kunming, China. He is a Lecturer with the School of Information Science and Engineering, Yunnan University, China. His research interests include natural language processing, text mining, and machine learning.



K. Robert Lai received the Ph.D. degree in computer science from North Carolina State University, Raleigh, NC, USA, in 1992. He is a Professor with the Department of Computer Science and Engineering, and the Director of Innovation Center for Big Data and Digital Convergence, Yuan Ze University, Taoyuan, Taiwan. His research interests include big data analytics, agent technologies, and mobile computing.

Xuejie Zhang received the Ph.D. degree in computer science and engineering from Chinese University of Hong Kong, Hong Kong, in 1998. He is a Professor in the School of Information Science and Engineering, and the Director of the High Performance Computing Center, Yunnan University, Kunming, China. His research interests include high-performance computing, cloud computing, and big data analytics.