SentiVec: Learning Sentiment-Context Vector via Kernel Optimization Function for Sentiment Analysis

Luyao Zhu, Wei Li, Yong Shi, *Senior Member, IEEE*, and Kun Guo

***Abstract*— Deep learning-based sentiment analysis (SA) methods have drawn more attention in recent years, which calls for more precise word embedding methods. This article proposes SentiVec, a kernel optimization function system for sentiment word embedding, which is based on two phases. The first phase is a supervised learning method, and the second phase consists of two unsupervised updating models, object-word-to-surrounding-words reward model (O2SR) and context-to-object-word reward model (C2OR). SentiVec is aimed at: 1) integrating the statistical information and sentiment orientation into sentiment word vectors and 2) propagating and updating the semantic information to all the word representations in a corpus. Extensive experimental results show that the optimal sentiment vectors successfully extract the features in terms of semantic and sentiment information, which makes it outperform the baseline methods on word similarity, word analogy, and SA tasks.**

***Index Terms*— Kernel function, natural language processing (NLP), sentiment vector, word embedding.**

I. Introduction

ONE of the key steps in natural language processing (NLP) tasks is to transform the unstructured and heterogeneous text information into structured data. This calls for word representation, which embeds words with real-valued vectors via probabilistic topic modeling or models semantic vector space models of language [1]. The vectors are the extracted features applied to subsequent NLP tasks, such as sentiment analysis (SA) [2], named entity recognition [3], parsing [4], question answering [5], and so on.

There are mainly two extension algorithms for word embedding in vector space models: 1) local context window approaches, like word2vec [6] and 2) global count-based approaches [7], like latent semantic analysis (LSA) [8]. However, these algorithms still have some limitations. Methods based on approaches 1) or 2) usually do not contain both global and local information at the same time. To address this problem, Pennington *et al.* [1] proposed GloVe, training word vectors by means of both global co-occurrence counts and local context window information. Nonetheless, this algorithm mainly concentrates on capturing semantic information instead of sentiment information among words. Some algorithms suffer from the computational complexity problem when the corpus size grows dramatically.

There are also several existing methods to construct sentiment vectors for textual SA [9], [10]. However, these methods are usually unable to get as good performances as the state-of-the-art methods on word similarity task.

Thus, we put forward SentiVec, a novel method based on kernel optimization function, to capture semantic similarity, statistical information, and sentiment information among words. First, this article analyzes the necessity of capturing the sentiment orientation of word vectors with respect to the cosine similarity of word vectors. Next, the word representations of sentiment words are trained via a supervised learning algorithm in the first phase of SentiVec. Then, the learned vectors are fed into an unsupervised local context window updating algorithm, consisting of object-word-to-surrounding-words reward model (O2SR) and context-to-object-word reward model (C2OR), to learn sentiment vectors for both sentiment and nonsentiment words in the second phase. The proposed method achieves better performances on word similarity, word analogy, and SA tasks than the state-of-the-art methods word2vec [6] and Glove [1].

The main contributions of this article are shown as follows.

1) Introducing the kernel function into an optimization function to learn sentiment vectors for SA, in which continuous sentiment polarity scores rather than discrete points are used during training process.
2) Efficiently leveraging a global optimization model to integrate global statistical information, sentiment similarity, and sentiment polarity scores into the vectors for sentiment words.
3) Proposing two local context window iterative updating models O2SR and C2OR for capturing sentiment similarity and semantic information among both sentiment and nonsentiment words.

The rest of this article is organized as follows. In Section II, we introduce related work pertaining to word embedding and sentiment vectors. In Section III, we elaborate the concept of

Manuscript received August 6, 2018; revised November 27, 2019 and March 26, 2020; accepted June 28, 2020. This work was supported by the National Natural Science Foundation of China under Grant 71932008. *(Corresponding author: Wei Li.)*

Luyao Zhu and Wei Li are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: luyao001@e.ntu.edu.sg; wei008@e.ntu.edu.sg).

Yong Shi and Kun Guo are with the School of Economics and Management, University of Chinese Academy of Sciences, Beijing 100190, China.

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

Digital Object Identifier 10.1109/TNNLS.2020.3006531

SentiVec, global count-based models, and local context window updating algorithms in detail. In Section IV, we perform experiments and evaluations on proposed models. Our research work and future research plan are concluded in Section V.

II. Related Work

A. Word Embedding

Existing word embedding algorithms are primarily based on two model families: 1) probabilistic topic modeling and 2) vector space models [11].

Classical probabilistic topic modeling is the latent Dirichlet allocation (LDA)-related model family. LDA was proposed by Blei *et al.* [12], which assumed that every document was the combination of latent topics. There are also several extensions of LDA in later research. However, these word vectors may not be as sensible as points in a k-dimensional space since LDA concentrates on topics instead of word meanings. Some other LDA-based work [13]–[15] captures sentiment besides topics. But these models emphasize sentiment-related topics rather than embedding word in vector spaces.

As mentioned in Section I, there are mainly three model families in vector space models: 1) global count-based methods; 2) local context window methods; and 3) hybrid methods. The first model family is the global count-based method. For example, LSA [8] is probably the best-known vector space model in the first model family. LSA aims to embed words directly [16]. It applies singular value decomposition (SVD) to factorize a term-document co-occurrence matrix to learn semantic word representations [11]. Besides, two step canonical correlation analysis (TSCCA) [17], sparse random projections [18], and Hellinger principal component analysis (PCA) [19] are also global count-based methods capturing relevant information from co-occurrence statistics. These global methods efficiently utilize statistical information [1], but large quantities of choices (weights, dimensionality reduction algorithm, normalization) are needed to be determined with little theoretical guidance for researchers to follow [11]. However, the global count-based method, which is also a kind of statistical method, is generally semantically weak [20]. The second famous vector space model family is the local context window method trained on separate local context windows, which poorly leverages the statistical information of the corpus. Typical local context window methods include neural probabilistic language model proposed by Bengio [21], C&W [22], M&H [23], and word2vec [6]. These models use the context to predict the object words to learn word vectors through the combination of language models and neural networks. Among the second models' families, word2vec is prevalent in NLP tasks and has a better performance. The third model family is hybrid model. Hybrid models train on a local context window and utilize the global statistics at the same time. Among the models, GloVe [1] attracts more attention in recent research.

The aforementioned word embedding models are mainly built for general NLP tasks. Moreover, there are some researches focusing on task-oriented word embedding. AffectiveSpace model [24] is presented for emotive reasoning task and the consecutive model AffectiveSpace2 [25] is tailored for concept-level SA. These models perform well on SA-related tasks and inspire the later research work on sentiment vector model.

B. Sentiment Vector

The word representation models are put forward for multiple NLP tasks in the beginning. Later in order to improve the performance on SA, some research articles gradually focus on integrating sentiment information into the word vector, which is called sentiment vector.

Maas *et al.* [11] presented a probabilistic model capturing semantic and sentiment similarity at the same time. Tang *et al.* [26] proposed sentiment-specific word embedding (SSWE) for twitter sentiment classification based on C&W. Saif *et al.* [10] put forward SentiCircles for SA on twitter. And there is also some work introducing the concept and rule-based representation method of specific sentiment vector for SA, such as [9] and [27]. However, these methods are usually not suitable for word relatedness evaluation and some methods are even not qualified for unsupervised learning.

Recently, some research articles proposed an overall framework for embedding based on deep learning methods. Cambria *et al.* [28] employed multilayer neural network for target word representation and bidirectional long short-term memory (BiLSTM) for context embedding. In addition, some articles focused on extending existing methods. Shi *et al.* [29] proposed a model for learning domain-specific and sentiment aware word embeddings based on word2vec extensions. Fu *et al.* [30] put forward SWpredict and SWRank for sentiment word embedding based on word2vec extensions. Li *et al.* [31] proposed a GloVe-based method DLJT2 for learning sentiment vector. Yu *et al.* [32] presented a refinement model applicable to pretrained word vectors, such as word2vec and GloVe, for sentiment embeddings. The models listed above usually obtain good performances on specific sentiment classification tasks. However, these methods analyze the learned embeddings mainly based on sentiment classification tasks and lack the intrinsic evaluation of word embeddings, e.g., word relatedness evaluation, word analogy, and so on.

Thus, this article proposes a semisupervised word embedding method incorporating semantic information, sentiment orientation, and statistical information into word vectors. Our method is designed as a combination of local context method and global statistics. It aims to gain good performance on word relatedness evaluation, word analogy, and sentiment classification tasks.

III. Mechanism of SentiVec Model

Our SentiVec model is tailored for learning sentiment-context vectors discussed in Section III-A. The model and algorithms, based on two phases, are elaborated in Sections III-B and III-C. The first phase proposes a supervised learning method (also a global count-based method) for the embedding of sentiment words, capturing sentiment information and similarities. This phase requires word occurrences and calibrated sentiment scores and uses the adaptive stochastic gradient descent to solve the

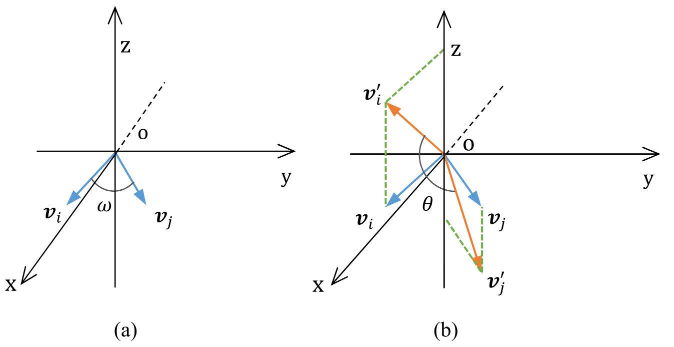

Fig. 1. Changes of the intersection angle in different dimensional space. (a) Preimage in low-dimensional space. (b) Image in high-dimensional space.

objective function. The second phase is an unsupervised learning method (also a local context window method) for learning sentiment-context vectors of both sentiment words and nonsentiment words, which is designed for learning contextual semantic information and sentiment similarities among all words in the corpus.

A. Sentiment-Context Vector

In this section, we build a sentiment vector (sentiment-context vector) capable of describing sentiment orientations and similarities as well as semantic similarities among words.

It shows that the correlation between the sentiment similarities and semantic relatedness is not significant sometimes. For example, in sentence "the buffet breakfast is good/bad," sentiment words "good" and "bad" are totally opposite in sentiment orientation, although they may be very close in semantic relatedness. This means the intersection angle between the word vectors tends to be π in terms of sentiment orientation and the one in terms of semantic similarity should be less than $(\pi/2)$. This means it is impossible to treat the two intersection angles as one in the same vector space. And it shows that if we consider the image vectors after the nonlinear transformation in the higher dimensional space in terms of sentiment information and consider the preimage vectors in the lower dimensional space in terms of semantic information, the aforementioned dilemma will be solved.

It is assumed that, given word w_i, the word vector $\boldsymbol{v}_i \in \mathbb{R}^d$, $\boldsymbol{v}'_t \in \mathbb{R}^m$, where d and m are the dimensions and $d < m$. As shown in Fig. 1, the sentiment-context vector for word w_i is a 3-D vector $\boldsymbol{v'_i} = (v'_{i1}, v'_{i2}, v'_{i3})$, where v'_{i1}, and v'_{i2} depict the contextual semantic information of word w_i, and v'_{i3} depicts the sentiment information of word w_i. Thus, $\boldsymbol{v'_i}$ is the image vector in a higher dimensional space in terms of sentiment information as well as semantic information. And $\boldsymbol{v_i} = (v_{i1}, v_{i2})$ is the preimage in a lower dimensional space in terms of semantic information. Similarly, we could assume a sentiment-context vector for word w_j, with $\boldsymbol{v'_j} = (v'_{j1}, v'_{j2}, v'_{j3})$ as an image vector in a higher dimensional space in terms of sentiment information as well as semantic information. And it is also assumed $\boldsymbol{v_j} = (v_{j1}, v_{j2})$ is the preimage in a lower dimensional space in terms of semantic information.

In special cases, as shown in Fig. 1, the intersection angle ω between preimage vectors $\boldsymbol{v_i}$ and $\boldsymbol{v_j}$ is significantly less than $(\pi/2)$, which indicates the cosine similarity $s(\boldsymbol{v_i}, \boldsymbol{v_j}) = (\boldsymbol{v_i^T} \boldsymbol{v_j}/\|\boldsymbol{v_i}\|\|\boldsymbol{v_j}\|) > 0$. After nonlinear transformation $\varphi : \mathbb{R}^d \to \mathbb{R}^m$, when $d = 2$ and $m = 3$, the intersection angle θ between image vectors $\boldsymbol{v'_i}$ and $\boldsymbol{v'_j}$ is significantly larger than $(\pi/2)$, which means the cosine similarity $s(\varphi(\boldsymbol{v_i}), \varphi(\boldsymbol{v_j})) = (\varphi(\boldsymbol{v_i})^{\boldsymbol{T}} \varphi(\boldsymbol{v_j})/\|\varphi(\boldsymbol{v_i})\|\|\varphi(\boldsymbol{v_j})\|) < 0$. This means, considering both the sentiment information and context semantic information, the cosine similarity will change compared with the original case where only contextual semantic information is given.

However, simply introducing sentiment information, such as sentiment scores, into one more vector component is not rigorous in mathematics and it cannot ensure that the positive and negative vectors are linearly separable in a high-dimensional space.

Thus, one of the key goals is to find a qualified nonlinear transformation $\varphi : \mathbb{R}^d \to \mathbb{R}^m$, meeting the aforementioned requirements related to contextual semantic and sentiment information, so that the image vectors containing different sentiment information are linearly separable in a high-dimensional feature space. To accomplish this goal, two tasks have to be achieved: 1) reducing the computation complexity and 2) reducing the distortion of sentiment information while updating the word vectors in low-dimensional feature space.

1) Reducing the Computation Complexity: Given such a qualified specific nonlinear transformation φ, all the preimage vectors are mapped into a high-dimensional feature space $\mathbb{R}^m$ from a low-dimensional input space $\mathbb{R}^d$. But this will lead to the curse of dimension as d and m increase.

Considering the distance between the two image vectors $\varphi(\boldsymbol{v_i})$ and $\varphi(\boldsymbol{v_j})$ in high-dimensional space, $d(\varphi(\boldsymbol{v_i}), \varphi(\boldsymbol{v_j})) = \|\varphi(\boldsymbol{v_i}) - \varphi(\boldsymbol{v_j})\| = \langle\varphi(\boldsymbol{v_i}), \varphi(\boldsymbol{v_i})\rangle - \boldsymbol{2}\langle\varphi(\boldsymbol{v_i}), \varphi(\boldsymbol{v_j})\rangle + \langle\varphi(\boldsymbol{v_j}), \varphi(\boldsymbol{v_j})\rangle$. If both the two image vectors have the same norm, $\langle\varphi(\boldsymbol{v_i}), \varphi(\boldsymbol{v_j})\rangle$, i.e., the similarity (inner product) of $\varphi(\boldsymbol{v_i})$ and $\varphi(\boldsymbol{v_j})$, becomes the only factors influencing the distance. In this case, different input vectors $\boldsymbol{v_i}$, $\boldsymbol{v_j}$ could be embedded into different clusters in high-dimensional space through controlling the inner product between their image vectors $\varphi(\boldsymbol{v_i})$, $\varphi(\boldsymbol{v_j})$ by rule. And if the inner product $\langle\varphi(\boldsymbol{v_i}), \varphi(\boldsymbol{v_j})\rangle$ could be obtained through computation without knowing the nonlinear transformation φ, this will reduce much more computation complexity and alleviate the curse of dimension.

According to [33], kernel-based algorithms are elegant in the sense that the "kernel trick" [34] enables the algorithm to operate implicitly in very high- (sometimes infinite-) dimensional feature spaces without explicitly knowing the actual feature space representation φ. That is, it simplifies the computation complexity through $K(\boldsymbol{v_i}, \boldsymbol{v_j}) = \langle\varphi(\boldsymbol{v_i}), \varphi(\boldsymbol{v_j})\rangle$, where $K(\boldsymbol{v_i}, \boldsymbol{v_j})$ is the kernel function. More generally, kernel function is defined as $K(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+ \cup \{0\}$. Moreover, the Gaussian or radial basis function (RBF) kernel [35] induces an infinite dimensional feature space [36], where all image vectors, with the same norm, become orthogonal when they are distant in the input space with respect to the scale parameter σ^2. Here, the Gaussian kernel is represented as $K(\boldsymbol{v_i}, \boldsymbol{v_j}) = \exp(-(\|\boldsymbol{v_i} - \boldsymbol{v_j}\|^2/\sigma^2))$. Therefore, Gaussian kernel is a good choice to solve the first problem.

2) Reducing the Distortion of Sentiment Information While Updating the Word Vectors in Low-Dimensional Feature Space: In our model, if we preset a benchmark vector $\boldsymbol{o}$, we can easily represent the intrinsic sentiment scores of the other sentiment vector $\boldsymbol{v_i}$ via $s(\varphi(\boldsymbol{v_i}), \varphi(\boldsymbol{o})) = K(\boldsymbol{v_i}, \boldsymbol{o})$, where the range of the sentiment scores is [−1,1]. The basic benchmark vector $\boldsymbol{o}$ is usually the sentiment vector with the most positive or negative sentiment score. Based on these conditions, the word embedding $\boldsymbol{v_i}$ for sentiment word w_i could be generated through training $K(\boldsymbol{v_i}, \boldsymbol{o})$ toward a manually labeled sentiment score. But the word vectors containing only sentiment information are not capable of representing contextual semantic information. In this case, the word embedding may be updated during the process of capturing semantic information into vectors. And it would be better if the sentiment information would not be distorted during updating procedure.

For all translation-invariant kernels, it holds that $K(\boldsymbol{v_i} + \boldsymbol{\gamma}, \boldsymbol{o} + \boldsymbol{\gamma}) = K(\boldsymbol{v_i}, \boldsymbol{\gamma})$. Thus, if $\boldsymbol{\gamma}$, which is the variation of vector $\boldsymbol{v_i}$, in updating process is known, a new benchmark vector $\boldsymbol{o}+\boldsymbol{\gamma}$ could be obtained. Then, the dot product of $\boldsymbol{v_i} + \boldsymbol{\gamma}$ and $\boldsymbol{o} + \boldsymbol{\gamma}$ after updating is the same as that of $\boldsymbol{v_i}$ and $\boldsymbol{o}$ in high-dimensional space. This means that the sentiment information of sentiment word w_i will not be distorted, since $K(\boldsymbol{v_i}+\boldsymbol{\gamma}, \boldsymbol{o}+\boldsymbol{\gamma}) = K(\boldsymbol{v_i}, \boldsymbol{o})$. In this case, the Gaussian kernel, as a translation-invariant kernel, is a good choice to solve the second problem as well.

According to the aforementioned analysis, the translation-invariant Gaussian kernel with the same norm for all vectors and capacity for computation complexity reduction is the suitable choice for solving the aforementioned problems in this article.

B. Capturing Sentiment Information

A probabilistic model of the scores of sentiment words is built based on a continuous mixture distribution over sentiment words. It is assumed that the sentiment scores $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)$ are subject to multivariate probability distribution. Then, some notations are introduced. Given a sentiment lexicon D, $N = |D|$, $y_i \in \mathbb{R}$ is the labeled normalized sentiment score of word w_i, so y_i is an observed value. $\boldsymbol{y}$ is an N-dimensional vector for labeled sentiment scores, where N is the number of sentiment words. V is the matrix of word vectors and $V \in \mathbb{R}^{(N\times d)}$, where d is the dimension of the vectors. It is assumed that the intrinsic sentiment score s_i of word w_i can be computed through $s_i = (K(\boldsymbol{v}_i, \boldsymbol{o})/(K(\boldsymbol{v}_i, \boldsymbol{v}_i))^{1/2}(K(\boldsymbol{o}, \boldsymbol{o}))^{1/2})$ according to Section III-A. In our model, the value of the basic benchmark vector $\boldsymbol{o}$ defined above is selected manually, so $\boldsymbol{s}(V)$ is the N-dimensional vector for intrinsic sentiment scores, where we suppose that $\boldsymbol{s}(V)$ is a function of variable V and $\boldsymbol{s}(V) = (s_1, s_2, \ldots, s_N)$. The probability of the sentiment scores is thus

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}, \boldsymbol{s}(V))d\boldsymbol{S}(V) = \int p(\boldsymbol{y}|\boldsymbol{s}(V))p(\boldsymbol{s}(V))d\boldsymbol{s}(V). \tag{1}$$

We define the conditional distribution $p(\boldsymbol{y}|\boldsymbol{s}(V))$ using a multivariate normal distribution function with parameters Σ and $\boldsymbol{s}(V)$. Σ is the covariance matrix, $\Sigma \in \mathbb{R}^{(N\times N)}$. $\boldsymbol{s}(V)$ is the mean vector of multiple random variables. The conditional probability density function of $\boldsymbol{y}$ is

$$\begin{aligned} &p(\boldsymbol{y}|\boldsymbol{s}(V)) \\ &= \frac{1}{(2\pi)^{\frac{N}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{s}(V))^T \Sigma^{-1}(\boldsymbol{y} - \boldsymbol{s}(V))\right). \end{aligned} \tag{2}$$

To reduce the number of unknown parameters, we use $\hat{\Sigma}$ as the estimate of Σ. Note that if Σ is an identity matrix, each component of the vector $\boldsymbol{y}$ is conditionally independent. In this work, it is assumed that the components of the vector $\boldsymbol{y}$ are mutually independent, for it is supposed that the annotations of sentiment words are conditionally independent.

Instead of simply assigning an estimate of $p(\boldsymbol{s}(V))$ in (1), we assume a multivariate normal distribution prior on $\boldsymbol{s}(V)$, which is also called the Gaussian prior. It is supposed that the intrinsic sentiment orientations of the sentiment words are usually different from zero and explicitly positive or negative, thus each component of the mean vector is assumed to be proportional to its observation sample y_k which is the corresponding component of vector $\boldsymbol{y}$. The probability density function of $\boldsymbol{s}(V)$ is

$$\begin{aligned} &p(\boldsymbol{s}(V)) \\ &= \frac{1}{(2\pi)^{\frac{N}{2}}|\Sigma_0^{-1}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{s}(V) - \delta\boldsymbol{y})^{\mathrm{T}} \Sigma_0^{-1}(\boldsymbol{s}(V) - \delta\boldsymbol{y})\right) \end{aligned} \tag{3}$$

where Σ_0 is a covariance matrix and $\Sigma_0 \in \mathbb{R}^{(N\times N)}$. We use $\hat{\Sigma}_0$ as an estimate of Σ_0 according to the corpus, which is illustrated in Section IV. δ is a scalar very close to zero.

Then, a maximum likelihood learning for our model is derived, given a set of calibrated sentiment lexicons and unlabeled documents. The model is designed to search for the best estimate of $\boldsymbol{s}(V)$ for the maximum of $p(\boldsymbol{s}(V)|\boldsymbol{y})$. Thus, the learning problem is

$$\begin{aligned} \max_V p(\boldsymbol{s}(V)|\boldsymbol{y}) &= \max_V \frac{p(\boldsymbol{y}, \boldsymbol{s}(V))}{p(\boldsymbol{y})} \\ &= \max_V \frac{p(\boldsymbol{y}|\boldsymbol{s}(V))p(\boldsymbol{s}(V))}{\int p(\boldsymbol{y}|\boldsymbol{s}(V))p(\boldsymbol{s}(V))d\boldsymbol{s}(V)} \\ &\propto \max_V p(\boldsymbol{y}|\boldsymbol{s}(V))p(\boldsymbol{s}(V)) \end{aligned} \tag{4}$$

which is the maximum *a posteriori* (MAP) problem. According to the monotonicity of the $\ln(x)$ function, we simplify the learning problem as

$$\begin{aligned} \max_V p(\boldsymbol{s}(V)|\boldsymbol{y}) &\propto \max_V \ln(p(\boldsymbol{y}|\boldsymbol{s}(V))p(\boldsymbol{s}(V))) \\ &\propto \min_V \left\{ \begin{aligned} &\frac{1}{2}(\boldsymbol{y} - \boldsymbol{s}(V))^{\mathrm{T}} \Sigma^{-1}(\boldsymbol{y} - \boldsymbol{s}(V)) \\ &+\frac{1}{2}(\boldsymbol{s}(V) - \delta\boldsymbol{y})^{\mathrm{T}} \Sigma_0^{-1}(\boldsymbol{s}(V) - \delta\boldsymbol{y}) \end{aligned} \right\}. \end{aligned} \tag{5}$$

In this work, to simplify the optimization function, we set $\|\varphi(\boldsymbol{v}_i)\| = (K(\boldsymbol{v}_i, \boldsymbol{v}_i))^{1/2} = 1$. Besides, Σ is assumed to be an identity matrix. Therefore, the optimization function can be written as

$$\min_{V}\left\{\begin{array}{l}\frac{1}{2}\sum_{i=1}^{n}(K(\boldsymbol{v}_i,\boldsymbol{o})-y_i)^2\\ +\sum_{i=1}^{n}\sum_{j\neq i}A_{i,j}[(K(\boldsymbol{v}_i,\boldsymbol{o})-\delta_i y_i)-(K(\boldsymbol{v}_j,\boldsymbol{o})-\delta_j y_j)]^2\end{array}\right\}$$

$$\mathrm{s\cdot t\cdot}\sqrt{K(\boldsymbol{v}_i,\boldsymbol{v}_i)}=1\quad\forall i$$

$$\min_{V}\left\{\begin{array}{l}\frac{1}{2}\sum_{i=1}^{n}(K(\boldsymbol{v}_i,\boldsymbol{o})-y_i)^2\\ +\frac{\alpha}{2}\sum_{i=1}^{n}\sum_{j\neq i}A_{i,j}[(K(\boldsymbol{v}_i,\boldsymbol{o})-\delta_i y_i)-(K(\boldsymbol{v}_j,\boldsymbol{o})-\delta_j y_j)]^2\\ +\beta\sum_{i=1}^{n}\lambda_i(\sqrt{K(\boldsymbol{v}_i,\boldsymbol{v}_i)}-1)\end{array}\right\} \tag{6}$$

where $\Sigma_0^{-1}=L=D-A$, L is a Laplacian matrix, D is a degree matrix, and A is an adjacency matrix. α and β are the hyper-parameters and λ_i is the Lagrange multiplier. In this article, we adopt the Gaussian kernel function to compute the inner product between nonlinear transformed $\varphi(\boldsymbol{v}_i)$ and $\varphi(\boldsymbol{v}_j)$ for word w_i and w_j since for any vector $\boldsymbol{v}_i, K(\boldsymbol{v}_i,\boldsymbol{v}_i)=1$ in the Gaussian kernel function. For other shift-invariant kernels such as the Laplacian kernel and the Cauchy kernel, the formula $K(\boldsymbol{v}_i,\boldsymbol{v}_i)=1,\forall t$ also holds. Thus, the final optimization function is

$$\arg\min_{V}\times\left\{\begin{array}{l}\frac{1}{2}\sum_{i=1}^{N}(K(\boldsymbol{v}_i,\boldsymbol{o})-y_i)^2\\ +\frac{\alpha}{2}\sum_{i=1}^{N}\sum_{j\neq i}A_{i,j}[(K(\boldsymbol{v}_i,\boldsymbol{o})-\delta_i y_i)-(K(\boldsymbol{x}_j,\boldsymbol{o})-\delta_j y_j)]^2\end{array}\right\} \tag{7}$$

where

$$K(\boldsymbol{x}_i,\boldsymbol{o})=\exp\left(-\frac{\|\ \boldsymbol{v}_i-\boldsymbol{o}\ \|^2}{2\sigma^2}\right).$$

And

$$\boldsymbol{o}=\begin{cases}\mathbf{1}, & \text{if } y_i\in R_+\\ -\mathbf{1}, & \text{if } y_i\in R_-\end{cases}$$

since the range of $K(\mathbb{R}^m,\mathbb{R}^m)\subseteq\mathbb{R}^+\cup\{0\}$. We use the sentiment score of sentiment word w_i in the sentiment lexicon as y_i.

C. Capturing Semantic Information and Propagating Sentiment Orientation

After solving the final optimization function for V in the first phase, we consider the propagation of sentiment orientation to other word vectors beyond the sentiment lexicon and capture the semantic information of all the words in the training corpus. In the second phase, two local context window algorithms, O2SR and C2OR, are proposed to update the vectors for both sentiment words and nonsentiment words. The objective functions of both algorithms are devised to maximize the expected reward, combined with negative sampling. The updating algorithms, of which the detailed models are elaborated in Sections III-C1 and III-C2, respectively, are capable of integrating sentiment similarity and semantic information for all the training vectors. And the pseudo-codes of both algorithms are shown in Sections III-C1 and III-C2.

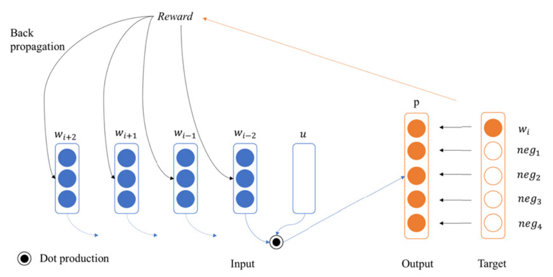

Fig. 2. O2SR architecture.

1) Object-Word-to-Surrounding-Words Reward Model: In O2SR, the classifier predicts the surrounding word ($w\in\mathrm{Context}(w_c)$), given the object word (also called current word) w_c; then, it obtains reward $R.(\cdot)$ according to the classification result. Therefore, the objective of the model is to maximize the expected reward of predicting surrounding words based on the object word in the same sentence. This model is denoted as O2SR. It is elaborated in detail as follows. Fig. 2 is the illustration of the O2SR model.

Given a context window $(w_c,\mathrm{Context}(w_c))$, surrounding word $w\in\mathrm{Context}(w_c)$, and a corresponding negative sample set U_w, we assume a sigmoid model for $p(u|w)$, the probability of predicting word u given word w, where $u\in\{w_c\cup U_w$. We use $\boldsymbol{v}_w$ as the word representation of surrounding word w and assume that each word has an auxiliary vector $\boldsymbol{\theta}_w$ besides the sentiment vector

$$p(u\mid w)=\begin{cases}\sigma\left(\boldsymbol{v}_w^{\mathrm{T}}\boldsymbol{\theta}_u\right), & L_{w_c}(u)=1\\ 1-\sigma\left(\boldsymbol{v}_w^{\mathrm{T}}\boldsymbol{\theta}_u\right), & L_{w_c}(u)=0\end{cases} \tag{8}$$

where the label of the sample word u is

$$L_{w_c}(u)=\begin{cases}1, & \text{if } u=w_c\\ 0, & \text{if } u\neq w_c\end{cases}$$

and

$$\sigma\left(\boldsymbol{v}_w^{\mathrm{T}}\boldsymbol{\theta}_u\right)=\frac{1}{1+\exp(-\boldsymbol{v}_w^{\mathrm{T}}\boldsymbol{\theta}_u)}.$$

This is the key part for learning semantic information. As for sentiment information, we use $R_w(u)=\exp(-h_w(u))R.(w)$ as the sentiment reward for predicting word u given word w, where $h_w(u)=(-1)^{L_{w_c}(u)}\exp(-(\|\ \boldsymbol{v}_w-\boldsymbol{\theta}_u\ \|^2/2\sigma^2))$, $R.(w)$ is the latest sentiment reward obtained from predicting word w given a certain word. Therefore, $H_w(u)=(r_w(u)+1):=\exp(-h_w(u))$ is the total return and $r_w(u)$ is the linear return.

The expected sentiment reward is derived as

$$\begin{aligned}E(w_c, w, u) &= p(u \mid w)R_w(u)\\ &= \sigma\left(\boldsymbol{v}_w^{\mathrm{T}}\boldsymbol{\theta}_u\right)^{L_{w_c}(u)}\left(1-\sigma\left(\boldsymbol{v}_w^{\mathrm{T}}\boldsymbol{\theta}_u\right)\right)^{1-L_{w_c}(u)}\\ &\quad \times \exp(-h_w(u))R_{\cdot}(w).\end{aligned} \tag{9}$$

For the vectors in the whole corpus C, the log likelihood reward function is defined as

$$\begin{aligned}\mathcal{L} &= \sum_{w_c\in C}\ \sum_{w\in \mathrm{Contex}(w_c)}\ \sum_{u\in\{w_c\}\cup U_w} \ln E(w_c, w, u)\\ &= \sum_{w_c\in C}\ \sum_{w\in Contex(w_c)}\ \sum_{u\in\{w_c\}\}\cup U_w}\\ &\quad\times\Bigg\{L_{w_c}(u)\ln\left[\sigma\left(\boldsymbol{v}_w^{T}\boldsymbol{\theta}_u\right)\right]\\ &\quad + [1-L_{w_c}(u)]\\ &\quad \times \ln\left[1-\sigma\left(\boldsymbol{v}_w^{\mathrm{T}}\boldsymbol{\theta}_u\right)\right]\\ &\quad + (-1)^{L_{w_c}(u)+1}\exp\left(-\frac{\|\ \boldsymbol{v}_w-\boldsymbol{\theta}_u\ \|^2}{2\sigma^2}\right)\Bigg\} + \ln R_0\end{aligned} \tag{10}$$

$$\begin{aligned}&\arg\max_V \mathcal{L}\\ &\propto \arg\max_V \arg \sum_{w_c\in C}\ \sum_{w\in Contex(w_c)}\ \sum_{u\in\{w_c\}\cup U_w}\\ &\quad\times\Bigg\{L_{w_c}(u)\ln\left[\sigma\left(\boldsymbol{v}_w^{\mathrm{T}}\boldsymbol{\theta}_u\right)\right]\\ &\quad + [1-L_{w_c}(u)]\ln\left[1-\sigma\left(\boldsymbol{v}_w^{\mathrm{T}}\boldsymbol{\theta}_u\right)\right]\\ &\quad + (-1)^{L_{w_c}(u)+1}\exp\left(-\frac{\|\ \boldsymbol{v}_w-\boldsymbol{\theta}_u\ \|^2}{2\sigma^2}\right)\Bigg\}\end{aligned} \tag{11}$$

where R_0 is the original(first) return of predicting the first word in corpus C. In (11), the objective of the O2SR model is also to maximize the total return of prediction. $\mathcal{L}(w_c, w, u)$ is written as $\ln E(w_c, w, u)$, then the gradient of $\mathcal{L}(w_c, w, u)$ can be written as

$$\begin{aligned}&\frac{\partial\mathcal{L}(w_c, w, u)}{\partial\boldsymbol{\theta}_u}\\ &= [L_{w_c}(u)-\sigma(\boldsymbol{v}_w^{\mathrm{T}}\boldsymbol{\theta}_u)]\boldsymbol{v}_w\\ &\quad + \frac{(-1)^{L_{w_c}(u)}}{\sigma^2}(\boldsymbol{\theta}_u-\boldsymbol{v}_w)\exp\left(-\frac{\|\ \boldsymbol{v}_w-\boldsymbol{\theta}_u\ \|^2}{2\sigma^2}\right)\end{aligned} \tag{12}$$

$$\begin{aligned}&\frac{\partial\mathcal{L}(w_c, w, u)}{\partial\boldsymbol{v}_w}\\ &= [L_{w_c}(u)-\sigma(\boldsymbol{v}_w^{\mathrm{T}}\boldsymbol{\theta}_u)]\boldsymbol{\theta}_u\\ &\quad + \frac{(-1)^{L_{w_c}(u)}}{\sigma^2}(\boldsymbol{v}_w-\boldsymbol{\theta}_u)\exp\left(-\frac{\|\ \boldsymbol{v}_w-\boldsymbol{\theta}_u\ \|^2}{2\sigma^2}\right).\end{aligned} \tag{13}$$

2) Context-to-Object-Word Reward Model: In C2OR, the classifier predicts the object word w_c, given the context Context(w_c); then, it obtains reward $R_{\cdot}(\cdot)$ according to the classification result. Therefore, the objective of the model is to maximize the expected reward of predicting the objective word based on the context in the same sentence. This model is denoted as C2OR. Using the notations in Section III-C1, the model is elaborated in detail as follows. Fig. 3 is the illustration of the C2OR model.

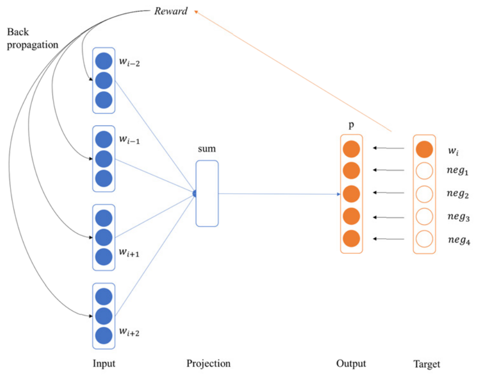

Fig. 3. C2OR architecture.

Given a context window $(w_c, \mathrm{Context}(w_c))$, surrounding word $w \in \mathrm{Context}(w_c)$ and a corresponding negative sample set U_{w_c}, we assume a sigmoid model for $p(u|\mathrm{Context}(w_c))$, the probability of predicting word u given context Context(w_c), where $u \in \{w_c \cup U_{w_c}$. We use $\boldsymbol{x}_{w_c} = \sum_{w\in\mathrm{Context}(w_c)}\boldsymbol{v}_w$ as the word representation of the context for object word w_c and assume that each word has an auxiliary vector $\boldsymbol{\theta}$ besides the sentiment vector

$$p(u|\mathrm{Context}(w_c)) = \begin{cases}\sigma\left(\boldsymbol{x}_{w_c}^{\mathrm{T}}\boldsymbol{\theta}_u\right), & L_{w_c}(u)=1\\ 1-\sigma\left(\boldsymbol{x}_{w_c}^{\mathrm{T}}\boldsymbol{\theta}_u\right), & L_{w_c}(u)=0\end{cases} \tag{14}$$

where the label of the sample word u is

$$L_{w_c}(u) = \begin{cases}1, & \text{if } u = w_c\\ 0, & \text{if } u \neq w_c\end{cases}$$

and $\sigma(\boldsymbol{x}_{w_c}^{\mathrm{T}}\boldsymbol{\theta}_u) = (1/1+\exp(-\boldsymbol{x}_{w_c}^{\mathrm{T}}\boldsymbol{\theta}_u))$. This is the key part for learning semantic information. As for sentiment information, we use $R_{w_c}(u) = \exp(-h_{w_c}(u))R_{\cdot,t-1}$ as the sentiment reward of predicting word u given context Context(w_c), where $h_{w_c}(u) = (-1)^{L_{w_c}(u)}\exp(-(\|\ \boldsymbol{v}_{w_c}-\boldsymbol{\theta}_u\ \|^2/2\sigma^2))$, $R_{\cdot,t-1}$ is the latest sentiment reward obtained from last prediction ($t-$ 1th prediction). Therefore, $H_{w_c}(u) = (r_{w_c}(u)+1) := \exp(-h_{w_c}(u))$ is the total return and $r_{w_c}(u)$ is the linear return. The expected sentiment reward is derived as

$$\begin{aligned}&E(w_c, \mathrm{Context}(w_c), u)\\ &= p(u \mid \mathrm{Context}(w_c))R_{w_c}(u)\\ &= \sigma\left(\boldsymbol{x}_{w_c}^{\mathrm{T}}\boldsymbol{\theta}_u\right)^{L_{w_c}(u)}\left(1-\sigma\left(\boldsymbol{x}_{w_c}^{\mathrm{T}}\boldsymbol{\theta}_u\right)\right)^{1-L_{w_c}(u)}\\ &\quad\cdot\exp(-h_{w_c}(u))R_{\cdot,t-1}.\end{aligned} \tag{15}$$

For the vectors in the whole corpus C, the log likelihood reward function is defined as

$$\begin{aligned}\mathcal{L} &= \sum_{w_c\in C}\ \sum_{u\in\{w_c\}\cup U_{w_c}} \ln E(w_c, \mathrm{Context}(w_c), u)\\ &= \sum_{w_c\in C}\ \sum_{u\in\{w_c\}\cup U_{w_c}}\left\{\begin{aligned}&L_{w_c}(u)\ln\left[\sigma\left(\boldsymbol{x}_{w_c}^{\mathrm{T}}\boldsymbol{\theta}_u\right)\right]\\ &+[1-L_{w_c}(u)]\ln\left[1-\sigma\left(\boldsymbol{x}_{w_c}^{\mathrm{T}}\boldsymbol{\theta}_u\right)\right]\\ &+(-1)^{L_{w_c}(u)+1}\exp\left(-\frac{\|\ \boldsymbol{v}_{w_c}-\boldsymbol{\theta}_u\ \|^2}{2\sigma^2}\right)\end{aligned}\right\}\\ &\qquad + \ln R_0\end{aligned} \tag{16}$$

$$\arg\max_{V} \mathcal{L} \propto \arg\max_{V} \sum_{w_c \in C} \sum_{u \in \{w_c\} \cup U_{w_c}} \times \left\{ \begin{array}{l} L_{w_c}(u) \ln\left[\sigma\left(\boldsymbol{x}_{w_c}^{\mathrm{T}} \boldsymbol{\theta}_u\right)\right] \\ +[1 - L_{w_c}(u)] \ln\left[1 - \sigma\left(\boldsymbol{x}_{w_c}^{\mathrm{T}} \boldsymbol{\theta}_u\right)\right] \\ +(-1)^{L_{w_c}(u)+1} \exp\left(-\frac{\| \boldsymbol{v}_{w_c} - \boldsymbol{\theta}_u \|^2}{2\sigma^2}\right) \end{array} \right\} \quad (17)$$

where R_0 is the original(first) return of predicting the first word in corpus C. In (17), the objective of the C2OR model is also to maximize the total return of prediction. $\mathcal{L}(w_c, \text{Context}(w_c), u)$ is written as $\ln E(w_c, \text{Context}(w_c), u)$; then, the gradient of $\mathcal{L}(w_c, \text{Context}(w_c), u)$ can be written as

$$\frac{\partial \mathcal{L}(w_c, \text{Context}(w_c), u)}{\partial \boldsymbol{\theta}_u} = \left[L_{w_c}(u) - \sigma\left(\boldsymbol{x}_{w_c}^{\mathrm{T}} \boldsymbol{\theta}_u\right)\right] \boldsymbol{x}_{w_c} + \frac{(-1)^{L_{w_c}(u)}}{\sigma^2} (\boldsymbol{\theta}_u - \boldsymbol{v}_{w_c}) \exp\left(-\frac{\| \boldsymbol{v}_{w_c} - \boldsymbol{\theta}_u \|^2}{2\sigma^2}\right) \quad (18)$$

$$\frac{\partial \mathcal{L}(w_c, \text{Context}(w_c), u)}{\partial \boldsymbol{x}_{w_c}} = \left[L_{w_c}(u) - \sigma\left(\boldsymbol{x}_{w_c}^{\mathrm{T}} \boldsymbol{\theta}_u\right)\right] \boldsymbol{\theta}_u + \frac{(-1)^{L_{w_c}(u)}}{\sigma^2} (\boldsymbol{v}_{w_c} - \boldsymbol{\theta}_u) \exp\left(-\frac{\| \boldsymbol{v}_{w_c} - \boldsymbol{\theta}_u \|^2}{2\sigma^2}\right). \quad (19)$$

3) Updating Algorithm of SentiVec in the Second Phase: As mentioned in the first paragraph of Section III-C above, the model in the first phase is tailored for learning word embedding for sentiment words. After solving the objective function in Section III-B, the vectors of sentiment words are fed to the algorithm in the second phase, which leads to the final result of SentiVec for both sentiment and nonsentiment words in the whole corpus via iterative random gradient descent updating.

The pseudo-code of the second phase is shown as follows.

IV. Experiment

A. Construction of SentiVec

1) First Phase: The famous SentiWordNet [37] is utilized as the source of calibrated sentiment scores. SentiWordNet measures the relative sentiment polarities of each word. That is, each word has a positive polarity and a negative polarity simultaneously, with calibrated scores ranging from zero to one on each polarity. The value of the score indicated the relative degree in each sentiment orientation (positive or negative). Besides, a word may have several records in the SentiWordNet since it may have different part of speech. For simplicity, we calculate the difference between two scores on positive and negative polarity to represent the absolute sentiment score of a word, where the sign of the difference stands for the polarity of a word and its absolute value indicates the degree in corresponding sentiment orientation. Then, we merge different records of a word. Finally, a positive sentiment lexicon with 3431 words and a negative lexicon with 5437 words are obtained.

Updating Algorithm for O2SR

```
Input $\boldsymbol{v}$ (vectors of sentiment words)
Random initialization for other vectors
FOR $w_c \in C$ DO
{
  FOR $w \in Contex(w_c)$ DO
  {
    $e = 0$
    FOR $u \in \{w_c \cup U_w$ DO
    {
      $l = \frac{(-1)^{L_{w_c}(u)}}{\sigma^2}(\boldsymbol{v}_w - \boldsymbol{\theta}_u)exp\left(-\frac{\|\boldsymbol{v}_w - \boldsymbol{\theta}_u\|^2}{2\sigma^2}\right)$
      $p = \sigma\left(\boldsymbol{v}_w^T \boldsymbol{\theta}_u\right)$
      $q = \eta(L_{w_c}(u) - p)$
      $e+ = q\boldsymbol{\theta}_u + l$
      $\theta_u+ = q\boldsymbol{v}_w - l$
    }
    $\boldsymbol{v}_w+ = e$
  }
}
```

Updating Algorithm for C2OR

```
Input $\boldsymbol{v}$ (vectors of sentiment words)
Random initialization for other vectors
FOR $w_c \in C$ DO
{
  $e = 0$
  $\boldsymbol{x}_{w_c} = \sum_{w \in Context(w_c)} \boldsymbol{v}_w$
  FOR $u \in \{w_c \cup U_{w_c}$ DO
  {
    $l = \frac{(-1)^{L_{w_c}(u)}}{\sigma^2}(\boldsymbol{v}_{w_c} - \boldsymbol{\theta}_u)exp\left(-\frac{\|\boldsymbol{v}_{w_c} - \boldsymbol{\theta}_u\|^2}{2\sigma^2}\right)$
    $p = \sigma\left(\boldsymbol{x}_{w_c}^T \boldsymbol{\theta}_u\right)$
    $q = \eta(L_{w_c}(u) - p)$
    $e+ = q\boldsymbol{\theta}_u + l$
    $\theta_u+ = q\boldsymbol{x}_{w_c} - l$
  }
  FOR $w \in Contex(w_c)$ DO
  {
    $\boldsymbol{v}_w+ = e$
  }
}
```

Next, after traversing the TripAdvisor[1] data set, statistical information of the word occurrence is extracted and stored in matrix B, then the word similarity matrix Σ_0 is generated through

$$\Sigma_{0,ij} = \begin{cases} \lg B_{ij}, & \text{if } B_{ij} > 0 \\ 0, & \text{otherwise.} \end{cases}$$

Fed with the sentiment lexicons and word similarity matrix, the optimization function is designed to optimize the sentiment-context vector for sentiment words in the vector space. In this phase, σ and α are set to 5 and 0.2, respectively.

2) Second Phase: To learn the contextual semantic information and sentiment similarities, our model was trained on approved large-size corpora, a 2018 Wikipedia dump with

[1]http://www.cs.virginia.edu/~hw5x/dataset.html

TABLE I
CONFIGURATION OF HYPER-PARAMETER

Hyper-parameter	Value
Window-size	5
Min-Count	50
Dimension	100
Number of thread	16
Learning rate	0.05
Corpus size	2.4B

2.4 billion tokens distributed in more than 4.4 million articles[2]. The quality of this corpora is fairly good since it is clean and all-inclusive. In this section, the hyper-parameter σ^2 is set to 585. The other key hyper-parameters are listed in Table I.

The hyper-parameters in both phases are selected according to the parameter analysis based on many repeated experiments on a small-scale data set. And the selection of hyper-parameter σ^2 in the Gaussian kernel is positively related to the dimension of input vectors.

B. Evaluation Methods

The sentiment-context vectors learned from our proposed SentiVec are evaluated on three NLP tasks, namely, word analogy, word similarity, and sentiment classification tasks.

1) Word Analogy: Word analogy task is devised to evaluate the learned relationships of word vectors, popularized by Mikolov *et al.* [38]. It contains many questions like "Beijing is to China as Paris is to __?." It is an exact answer if the result of this question is France. Furthermore, it is supposed that the word vectors generated from a vector space model are more probable to successfully describe the relationships of words in the real world if the model correctly answers as many such questions as possible from the evaluation data set[3]. The evaluation data set comprises 19 544 questions and is divided into semantic and syntactic subsets. The semantic questions are mainly analogies pertaining to places or people, like the aforementioned example. The syntactic questions are mainly analogies about forms of adjectives or verb tenses [1]. For example, "write is to writing as listen is to __?" SentiVec answers this question in the real world by finding a word whose word vector is closest to the result of the operation $\boldsymbol{v}_{\text{writing}} - \boldsymbol{v}_{\text{write}} + \boldsymbol{v}_{\text{listen}}$[4] from the perspective of the cosine similarity. Statistics of word analogy evaluation data set is listed in Table II.

2) Word Similarity: In addition, to directly evaluate the performance of SentiVec, we also perform the WordSim-353 [39] task and employ the cosine similarity to measure the similarity degree of word pairs. For instance, the cosine similarity of word pair "tiger & cat" is calculated by $(\boldsymbol{v}_{\text{tiger}}^{\text{T}}\boldsymbol{v}_{\text{cat}}/\|\boldsymbol{v}_{\text{tiger}}\|\|\boldsymbol{v}_{\text{cat}}\|)$. The human rating score ranges from 0 to 10, while the cosine similarity score ranges from -1 to 1. Therefore, we employ the Spearman correlation coefficient to measure the similarity between human rating sequence and cosine similarity sequence.

[2]https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2
[3]https://code.google.com/archive/p/word2vec/source/default/source
[4]W_i is the word vector of word i.

TABLE II
STATISTICS OF WORD ANALOGY DATA SET

Type	subdataset	account	Type	subdataset	account
	adjective → adverb	650		*capital → countries*	506
	opposite	240		*capital → world*	2415
Syntactic	*comparative*	992	Semantic	*currency*	54
	superlative	506		*city → state*	2265
	present → participle	812		*family*	380
	nationality → adjective	1371			
	past → tense	1406			
	plural	992			
	plural → verbs	600			
Sum		7569	Sum		5620

TABLE III
STATISTICS OF STANFORD SENTIMENT TREEBANK

Sentiment polarity label	Train	Dev	Test
1	1092	139	279
2	2218	289	633
3	1624	229	389
4	2322	279	510
5	1288	165	399
Total	8544	1101	2210

3) Sentiment Analysis: SA task is our basic focus since the SentiVec incorporates the sentiment information. SA refers to the use of text analysis and computational linguistics to identify and extract subjective information in source materials [40]–[42]. SA is extensively applied to product reviews and social media for a variety of applications, such as blogs [43], [44], Twitter [10], [45], and news [46]. It is aimed at analyzing users' sentiment polarities toward products, whereas opinion mining involves extracting users' opinions or orientations toward entities. There are two approach families for SA: 1) computational linguistic approaches, such as the lexicon-based SA methods [42], [47] and 2) the machine learning approaches, such as Naïve Bayes [48], [49], Bayesian network [50], neural networks [51], and support vector machine (SVM) [52]. SA, especially sentence-level sentiment polarity classification, is also applicable to evaluate the quality of word representations on sentiment information captured by different word embedding methods [53]. In this article, both the approved classifier SVM [54] and advanced neural network classifier convolutional neural network-long short term memory (CNN-LSTM) [55] are utilized to predict sentiment polarity of a review text. Given an input sentence $\text{Sen} = (w_1, w_2, \ldots, w_n)$ that contains n words, we use the average vector of all the word vectors to represent the sentence and then take the average vector as features to further feed the SVM classifier. Neural network classifier CNN-LSTM is capable of processing sequential data. Thus, we utilize the word vector of each word in Sen to feed the classifier in order. Fine-grained SA is conducted on Stanford Sentiment Treebank, a classic movie reviews data set [56]. Statistics of Stanford Sentiment Treebank data set is listed in Table III.

Besides, the proposed model is also compared with other word vector models containing sentiment and semantic information on a polarity data set introduced by Pang and Lee [48]. The data set is composed of 2000 movie reviews with binary sentiment polarity from the Internet movie database (IMDB). The models are evaluated by the SVM via LIBSVM [54], and the SVM parameters are the same as [48]. In this experiment section, models are trained on 25 000 labeled movie reviews from IMDB and 50 000 unlabeled movie reviews from IMDB [11][5]. Since all the word vectors in this section are of 50-D, the hyper-parameters of SentiVec are tuned accordingly during the training process. Here, in the second phase, the window size is set to 18, σ^2 is set to 3.5, the learning rate is set to 0.025, and the min-count is set to 15.

C. Baseline Models

The SentiVec is compared with the following word representation algorithms.

1) Word2vec: It has been the most famous and commonly used word representation technique since its release in 2013 by Mikolov *et al.* [6]. The algorithm has two variations, continuous bag of word (cbow) and skip gram (sg). It makes use of the local semantic information and gets the state-of-the-art results on several NLP tasks. The word2vec model was trained on the same corpus as SentiVec. Moreover, shared hyper-parameters stay the same as SentiVec.

2) GloVe: The model was presented by Pennington *et al.* [1] in 2014. It is an unsupervised learning algorithm for obtaining vector representations for words. Training is conducted on aggregated global word–word co-occurrence statistics from the aforementioned corpus, and the resulting representations showcase the linear substructures of the word vector space. GloVe exploits not only local semantic information but also global statistical information. Hence, it shows special superiority in certain cases. Similarly, shared hyper-parameters remain the same. The setting of some exclusive hyper-parameters follows the suggestion of GloVe research article. Therefore, we set $x_{\max} = 100$ and perform the experiment for 50 iterations since the dimension of the vectors is smaller than 300.

3) Hellinger Principal Component Analysis: Hellinger PCA (H-PCA) [19] is a word embedding algorithm using the word co-occurrence statistics and a well-known dimensionality reduction operation PCA [19][6]. It is concluded that, with an appropriate metric, this method can generate word embeddings as good as the ones generated by deep learning architectures. All the hyper-parameters are set as the authors suggested.

4) Word Vector Model (See [11]): The word vector model [11] captures sentiment content and semantic term-document information through unsupervised and supervised algorithm. The word vectors generated from this model are 50-D. The reported experiment result of this model in [11] is used in this article for comparison.

D. Experimental Results

1) Results of Word Analogy Task: As shown in Table IV, SentiVec(C2OR) outperforms all the benchmark methods on three indices. To be specific, it gains a slight superiority over benchmarks on semantic questions and a significant improvement on syntactic questions. The GloVe model is good at answering semantic questions but relatively weak in handling syntactic questions. Word2vec(cbow) gets balanced performances on both semantic and syntactic questions. Semantic questions refer to the analogies about people and places, while syntactic questions are primarily analogies with respect to forms of adjectives or verb tenses. People and places are mainly the relationships such as "capital–country," "capital–world," "country–currency," and "family (♂)–family (♀)." These relationships are quite stable in the corpus, especially a huge corpus. However, the meanings of adjectives are changeable in specific context. Therefore, all the models achieve better performances on semantic questions than syntactic questions.

TABLE IV
ACCURACY OF WORD ANALOGY

Model	semantic	syntactic	overall
H-PCA	0.4968	0.4721	0.4826
GloVe	0.7299	0.5706	0.6385
word2vec(sg)	0.6651	0.5702	0.6107
word2vec(cbow)	0.7180	0.6285	0.6666
SentiVec(C2OR)	**0.7306**	**0.6461**	**0.6821**

Best result is in bold.

TABLE V
SPEARMAN RANK CORRELATION ON WORD SIMILARITY TASK

Model	Technology	Corpus	Dimension	Accuracy
H-PCA[8]	--	IMDB	100	0.4137
H-PCA	--	2.4B	100	0.4722
word2vec	cbow	2.4B	100	0.5230
	sg	2.4B	100	0.5339
GloVe	--	2.4B	100	0.5208
SentiVec	O2SR	2.4B	100	0.5349
	C2OR	2.4B	100	**0.5379**

All vectors are 100-dimensional. Best result is in Bold.

2) Results of Word Similarity Task: Table V shows the results on word similarity task. The similarity scores are calculated by the cosine similarity between vector pairs. Then, we utilize Spearman's rank correlation coefficient to compute the sequence similarity between calculated cosine scores and human judgments. Using the Wikipedia data as the training corpus, SentiVec-based methods outperform other baselines and obtain the best results on WordSim-353 task. The sentiment-context vectors successfully describe the relations between word pairs, which shows the proposed model improves the training of word representations by a substantial margin. Moreover, H-PCA model has a result reported from IMDB data set[7]. Compared with this result, it is concluded that a large corpus contributes to the significant improvement of word embeddings. Therefore, a large corpus is necessary in order to train a high-quality word embeddings.

3) Results of SA Task: The results of the SA task are presented in Tables VI and VII. Given the confusion matrix,

[5]http://www.cs.cornell.edu/people/pabo/movie-review-data

[6]Source code URL: http://www.lebret.ch/words/

[7]http://www.andrew-maas.net/data/sentiment

TABLE VI
SENTIMENT POLARITY CLASSIFICATION RESULTS (SVM CLASSIFIER)

Model	P1	P2	P3	P4	P5	R1	R2	R3	R4	R5	F1	F2	F3	F4	F5	Acc.
H-PCA	0.000	0.386	0.000	0.283	0.000	0.000	0.528	0.000	**0.745**	0.000	N/A	0.446	N/A	**0.410**	N/A	0.323
GloVe	0.300	0.421	0.242	**0.331**	**0.470**	0.065	0.570	0.116	0.537	0.328	0.107	0.484	0.157	**0.410**	0.386	0.375
word2vec(sg)	**0.351**	**0.496**	0.219	0.297	**0.461**	**0.211**	0.107	**0.249**	**0.633**	**0.434**	**0.264**	0.176	**0.233**	**0.404**	**0.447**	0.326
word2vec(cbow)	0.318	0.416	0.211	0.314	0.431	0.122	0.542	0.090	0.449	**0.414**	0.176	0.471	0.126	0.370	0.422	0.365
SentiVec(O2SR)	0.305	0.369	**0.316**	**0.380**	0.257	0.154	**0.766**	0.108	0.261	0.361	0.205	**0.498**	0.161	0.309	0.300	**0.383**
SentiVec(C2OR)	**0.362**	**0.441**	**0.317**	0.328	0.459	**0.168**	**0.581**	**0.113**	0.482	0.411	**0.229**	**0.501**	**0.167**	0.390	**0.434**	**0.393**

P, R and F refer to the precision, recall and F-score, respectively. And they are calculated for each category. The bold font in a same column of the table means that the values are Top 2 best performances among all the methods.

TABLE VII
SENTIMENT POLARITY CLASSIFICATION RESULTS (CNN-LSTM CLASSIFIER)

Model	P1	P2	P3	P4	P5	R1	R2	R3	R4	R5	F1	F2	F3	F4	F5	Acc.
H-PCA	0.000	0.000	0.000	0.381	0.455	0.000	0.000	0.000	**0.999**	0.008	N/A	N/A	N/A	**0.552**	0.016	0.382
GloVe	0.000	0.000	**0.333**	0.383	0.472	0.000	0.000	0.006	**0.815**	0.339	N/A	N/A	0.012	0.521	0.395	0.402
word2vec(sg)	0.000	1.000	1.000	0.400	0.472	0.000	0.004	0.003	0.736	**0.490**	N/A	0.008	0.006	0.518	0.481	0.422
word2vec(cbow)	**0.500**	**0.300**	**0.333**	**0.408**	**0.526**	**0.011**	**0.032**	**0.019**	0.757	0.465	**0.022**	**0.058**	**0.036**	0.530	**0.494**	**0.437**
SentiVec(C2OR)	**0.500**	**0.448**	0.288	**0.445**	**0.595**	**0.041**	**0.110**	**0.116**	0.742	**0.572**	**0.076**	**0.177**	**0.165**	**0.556**	**0.583**	**0.475**

P, R and F refer to the precision, recall and F-score, respectively. And they are calculated for each category. The bold font in a same column of the table means that the values are Top 2 best performances among all the methods.

evaluation indices precision, recall and F1-score are calculated according to the following formulas [42]:

$$\text{Precision}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i} \tag{20}$$

$$\text{Recall}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i} \tag{21}$$

$$\text{F} - \text{score}_i = \frac{2\ \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}. \tag{22}$$

TP_i stands for true positive for the ith category and is equal to the number of labeled ith-category samples classified into ith category. FP_i represents false positive for the ith category and is equal to the number of labeled non-ith-category samples classified into ith category. FN_i is false negative for the ith category and is equal to the number of labeled ith-category samples classified into non-ith-category.

Table VI shows the precision, recall, F-score, and accuracy of baseline methods and SentiVec-based methods on fine-grained sentiment polarity classification of movie reviews. SentiVec-based methods significantly outperform all the benchmark methods in the overall trends since they successfully introduce sentiment information into the word vectors. H-PCA and GloVe models show a superiority in classifying the 4th category samples. Word2vec(cbow) is relatively weak because it lacks the global information, affecting the quality of word representations. Word2vec(sg) yields good performances on predicting the 1st, 3rd, and 5th category samples but gets the worst result in the 4th category prediction. Compared with other methods, SentiVec-based methods do well in classifying the 1st, 2nd, 3rd, and 5th category samples in general. In addition, the difference of 4th category samples classification performances between SentiVec and the best result is relatively slight. Importantly, SentiVec(C2OR) has a balanced performance on predicting movie reviews with different labels, making it an excellent model in encoding not only words but also sentences.

TABLE VIII
BINARY SENTIMENT POLARITY CLASSIFICATION RESULTS

Model	Window size	Dimension	Accuracy
word2vec(sg)	18	50	0.846
word2vec(cbow)	18	50	0.847
word vector [11]	--	50	0.84
SentiVec(O2SR)	18	50	**0.85**
SentiVec(C2OR)	18	50	**0.854**

All vectors are 50-dimensional. Best result is in Bold.

Similarly, Table VII shows that SentiVec (C2OR) makes a significant improvement in classifying movie reviews in the overall trend. Instead of using the average vectors of a sentence as the input features, advanced neural network classifier utilizes the features of each word in a sentence, which further demonstrates the superiority of the proposed word embedding model. H-PCA model gets good performances than any other model in terms of recall of the 4th category samples. GloVe model is relatively poor in predicting the 1st and 2nd category samples. Word2vec(cbow) has balanced performances on classifying movie reviews in general. Benefiting from the sentiment information, SentiVec(C2OR) increases the performances of fine-grained sentiment polarity prediction by a large margin.

A significance test is employed to further evaluate the superiority of the proposed model SentiVec(C2OR). All the models are performed ten times. Then, the result of paired-samples t-test demonstrates that SentiVec(C2OR) is better than benchmark methods.

Table VIII shows the binary sentiment polarity classification results on IMDB data set. The experiment result of word vector [11] is a reported result. Word vectors generated from [11] captures sentiment content and semantic information, which is likely to belong to a kind of sentiment-context vector described in this article. The results indicate that the SentiVec outperforms another word vector model [11] as well as word2vec on IMDB data set.

TABLE IX
ABLATION STUDY ON BINARY SENTIMENT POLARITY CLASSIFICATION

Model	Sentiment	Semantic	Win. size	Dim.	Acc.
SentiVec(1st Phase)	+	-	--	50	0.6265
O2SR(2nd Phase)	-	+	18	50	0.845
C2OR(2nd Phase)	-	+	--	50	0.8395
SentiVec(O2SR)	+	+	18	50	0.85
SentiVec(C2OR)	+	+	18	50	0.854

+ represents "Yes"; - represents "No". SentiVec (First Phase) refers to the word vectors generated from the first phase of our proposed model.

4) Ablation Study: To further explore the proposed SentiVec model, an ablation study is carried out on two main components of the SentiVec, i.e., the sentiment polarity information and contextual semantic information.

The experiments of the first phase, the second phase, and the complete SentiVec model are conducted separately on the IMDB data set. As shown in Table IX, the performance of the sole first phase degenerates dramatically in terms of classification accuracy. This is because the vectors generated from the first phase of SentiVec contains little semantic information, which leads to difficulty in document-level representation for classifier. Besides, the performances of the sole second phase also degrade in terms of classification accuracy, due to the lack of sentiment information.

The ablation study proves that the combination of the two phases integrates both the sentiment information and semantic information into the SentiVec and improves the experiment performance eventually.

V. CONCLUSION

In this article, we propose a novel framework SentiVec combining supervised and unsupervised techniques to learn high-quality word representations for both sentiment and non-sentiment words. SentiVec learns the sentiment vectors for sentiment words at the first phase and then incorporates semantic information into word representations for all words at the second phase via O2SR or C2OR. The evaluation experiment results show that the SentiVec improves the quality of word embedding through designing the kernel-based optimization function and updating algorithms. SA experiments conducted on Stanford Sentiment Treebank data set and IMDB data set explicitly indicate that the SentiVec outperforms baseline methods in predicting sentiment polarities. Extensive experiments, such as word similarity and word analogy, demonstrate that the proposed model is superior in depicting the semantic and syntactic information of word representations in vector space. The ablation study illustrates the combination of the first phase and the second phase of the proposed model assist to capture the sentiment and semantic information into SentiVec, which improves the performance on sentiment classification task. For future research, the influence of hyper-parameters should be discussed to provide insights regarding the proposed model.

REFERENCES

[1] 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.

[2] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1422–1432.

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

[4] R. Socher, J. Bauer, A. Y. Ng, and C. D. Manning, "Parsing with compositional vector grammars," in *Proc. 51st Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, Aug. 2013, pp. 455–465.

[5] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton, "Quantitative evaluation of passage retrieval algorithms for question answering," in *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2003, pp. 41–47.

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

[7] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2014, pp. 238–247.

[8] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Amer. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.

[9] P. Hui and M. Gregory, "Quantifying sentiment and influence in blogspaces," in *Proc. 1st Workshop Social Media Anal. (SOMA)*, 2010, pp. 53–61.

[10] H. Saif, Y. He, M. Fernandez, and H. Alani, "Contextual semantics for sentiment analysis of Twitter," *Inf. Process. Manage.*, vol. 52, no. 1, pp. 5–19, Jan. 2016.

[11] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2011, pp. 142–150.

[12] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[13] J. Boyd-Graber and P. Resnik, "Holistic sentiment analysis across languages: Multilingual supervised latent Dirichlet allocation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Oct. 2010, pp. 45–55.

[14] F. Li, M. Huang, and X. Zhu, "Sentiment analysis with global topics and local dependency," in *Proc. 24th AAAI Conf. Artif. Intell.*, 2007, pp. 1371–1376.

[15] C. Lin and Y. He, "Joint sentiment/topic model for sentiment analysis," in *Proc. 18th ACM Conf. Inf. Knowl. Manage. (CIKM)*, 2009, pp. 375–384.

[16] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *J. Artif. Intell. Res.*, vol. 37, pp. 141–188, Feb. 2010.

[17] P. Dhillon, J. Rodu, D. Foster, and L. Ungar, "Two step CCA: A new spectral method for estimating vector models of words," 2012, *arXiv:1206.6403*. [Online]. Available: http://arxiv.org/abs/1206.6403

[18] P. Li, T. J. Hastie, and K. W. Church, "Very sparse random projections," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2006, p. 287.

[19] R. Lebret and R. Collobert, "Word embeddings through hellinger PCA," in *Proc. EACL*, 2014, pp. 482–490.

[20] E. Cambria, "Affective computing and sentiment analysis," *IEEE Intell. Syst.*, vol. 31, no. 2, pp. 102–107, Mar. 2016.

[21] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[22] 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, Aug. 2011.

[23] A. Mnih and G. Hinton, "Three new graphical models for statistical language modelling," in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, 2007, pp. 641–648.

[24] E. Cambria, A. Hussain, C. Havasi, and C. Eckl, "AffectiveSpace: Blending common sense and affective knowledge to perform emotive reasoning," in *Proc. 1st Workshop Opinion Mining Sentiment Anal. (WOMSA)*, 2009, pp. 32–41.

[25] E. Cambria, J. Fu, F. Bisio, and S. Poria, "AffectiveSpace 2: Enabling affective intuition for concept-level sentiment analysis," in *Proc. Nat. Conf. Artif. Intell.*, 2015, pp. 508–514.

[26] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for Twitter sentiment classification," in *Proc. ACL*, 2014, pp. 1555–1565.

[27] L. Zhang, Y. Jia, B. Zhou, and Y. Han, "Detecting real-time burst topics in microblog streams: How sentiment can help," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 781–782.

[28] E. Cambria, S. Poria, D. Hazarika, and K. Kwok, "SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1795–1802.

[29] B. Shi, Z. Fu, L. Bing, and W. Lam, "Learning domain-sensitive and sentiment-aware word embeddings," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2018, pp. 2494–2504.

[30] P. Fu, Z. Lin, F. Yuan, W. Wang, and D. Meng, "Learning sentiment-specific word embedding via global sentiment representation," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 4808–4815.

[31] Y. Li, Q. Pan, T. Yang, S. Wang, J. Tang, and E. Cambria, "Learning word representations for sentiment analysis," *Cognit. Comput.*, vol. 9, no. 6, pp. 843–851, Dec. 2017.

[32] 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. 534–539.

[33] T.-T. Frieb and R. F. Harrison, "Perceptrons in kernel feature spaces," ACSE, Reston, VA, USA, Res. Rep. 720, 1998, pp. 1–16.

[34] B. Schölkopf, "The kernel trick for distances," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 301–307.

[35] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer, 1995, p. 140.

[36] A. Ruiz and P. E. López-de-Teruel, "Nonlinear kernel-based statistical pattern analysis," *IEEE Trans. Neural Netw.*, vol. 12, no. 1, pp. 16–32, Jan. 2001.

[37] S. Baccianella, A. Esuli, and F. Sebastiani, "SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," in *Proc. 7th Int. Conf. Lang. Resour. Eval. (LREC)*, 2010, pp. 2200–2204.

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

[39] L. Finkelstein *et al.*, "Placing search in context: The concept revisited," *ACM Trans. Inf. Syst.*, vol. 20, no. 1, pp. 116–131, 2002.

[40] K. Kim and J. Lee, "Sentiment visualization and classification via semi-supervised nonlinear dimensionality reduction," *Pattern Recognit.*, vol. 47, no. 2, pp. 758–768, Feb. 2014.

[41] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Found. Trends Inf. Retr.*, vol. 2, nos. 1–2, pp. 1–135, 2008.

[42] W. Li, K. Guo, Y. Shi, L. Zhu, and Y. Zheng, "DWWP: Domain-specific new words detection and word propagation system for sentiment analysis in the tourism domain," *Knowl.-Based Syst.*, vol. 146, pp. 203–214, Apr. 2018.

[43] D. E. O'Leary, "Blog mining-review and extensions: 'From each according to his opinion,'" *Decis. Support Syst.*, vol. 51, no. 4, pp. 821–830, Nov. 2011.

[44] J. Qi, X. Fu, and G. Zhu, "Subjective well-being measurement based on Chinese grassroots blog text sentiment analysis," *Inf. Manage.*, vol. 52, no. 7, pp. 859–869, Nov. 2015.

[45] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, "Sentiment analysis of Twitter data," in *Proc. Workshop Lang. Social Media*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 30–38.

[46] A. Devitt and K. Ahmad, "Sentiment polarity identification in financial news: A cohesion-based approach," in *Proc. 45th Annu. Meeting Assoc. Comput. Linguistics*, 2007, pp. 984–991.

[47] F. Wu, Y. Huang, Y. Song, and S. Liu, "Towards building a high-quality microblog-specific Chinese sentiment lexicon," *Decis. Support Syst.*, vol. 87, pp. 39–49, Jul. 2016.

[48] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proc. 42nd Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2004, pp. 271–278.

[49] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques," in *Proc. ACL-Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, vol. 10, 2002, pp. 79–86.

[50] X. Bai, "Predicting consumer sentiments from online text," *Decis. Support Syst.*, vol. 50, no. 4, pp. 732–742, Mar. 2011.

[51] L.-S. Chen, C.-H. Liu, and H.-J. Chiu, "A neural network based approach for sentiment classification in the blogosphere," *J. Informetrics*, vol. 5, no. 2, pp. 313–322, Apr. 2011.

[52] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics*, 2005, vol. 3, no. 1, pp. 115–124.

[53] T. Schnabel, I. Labutov, D. Mimno, and T. Joachims, "Evaluation methods for unsupervised word embeddings," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 298–307.

[54] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, 2011.

[55] W. Li, L. Zhu, Y. Shi, K. Guo, and E. Cambria, "User reviews: Sentiment analysis using lexicon integrated two-channel CNN-LSTM family models," *Appl. Soft Comput.*, vol. 94, Sep. 2020, Art. no. 106435.

[56] 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.

Luyao Zhu received the M.S. degree in management science and engineering from the University of Chinese Academy of Sciences, Beijing, China, in 2019. She is currently pursuing the Ph.D. degree in computer science and engineering with Nanyang Technological University, Singapore.

Her research interests include natural language processing, deep learning, and sentiment analysis.

Wei Li received the M.S. degree in management science and engineering from the University of Chinese Academy of Sciences, Beijing, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Nanyang Technological University, Singapore.

His main research interests include sentiment analysis, natural language processing, and deep learning. He is working on the construction of advanced deep learning models for sentiment analysis. He is also interested in question answering system.

Yong Shi (Senior Member, IEEE) received the Ph.D. degree in management science and computer system from the University of Kansas, Lawrence, KS, USA, in 1991.

He is currently a Professor with the Research Center on Fictitious Economy and Data Science, School of Economics and Management, University of Chinese Academy of Sciences, Beijing, China. He has authored more than 200 academic articles. His research interests include multiobjective programming, data mining, and machine learning.

Dr. Shi was awarded the 2013 Best Paper of the *Journal of Pattern Recognition*.

Kun Guo received the Ph.D. degree in management science and engineering from the University of Chinese Academy of Sciences, Beijing, China, in 2011.

She is currently an Associate Professor with the Research Center on Fictitious Economy and Data Science, School of Economics and Management, University of Chinese Academy of Sciences, Beijing, China. Her research interests include fictitious economy and complex systems.