Hindawi
Discrete Dynamics in Nature and Society
Volume 2021, Article ID 2822126, 12 pages
https://doi.org/10.1155/2021/2822126

Research Article

Graph Convolutional Network for Word Sense Disambiguation

Chun-Xiang Zhang, Rui Liu, Xue-Yao Gao, and Bo Yu

School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China

Correspondence should be addressed to Xue-Yao Gao; xueyao_gao@163.com

Received 16 May 2021; Revised 20 July 2021; Accepted 21 July 2021; Published 30 September 2021

Academic Editor: Akbar Ali

Word sense disambiguation (WSD) is an important research topic in natural language processing, which is widely applied to text classification, machine translation, and information retrieval. In order to improve disambiguation accuracy, this paper proposes a WSD method based on the graph convolutional network (GCN). Word, part of speech, and semantic category are extracted from contexts of the ambiguous word as discriminative features. Discriminative features and sentence containing the ambiguous word are used as nodes to construct the WSD graph. Word2Vec tool, Doc2Vec tool, pointwise mutual information (PMI), and TF-IDF are applied to compute embeddings of nodes and edge weights. GCN is used to fuse features of a node and its neighbors, and the softmax function is applied to determine the semantic category of the ambiguous word. Training corpus of SemEval-2007: Task #5 is adopted to optimize the proposed WSD classifier. Test corpus of SemEval-2007: Task #5 is used to test the performance of WSD classifier. Experimental results show that average accuracy of the proposed method is improved.

1. Introduction

In natural language, there is phenomenon that a polysemous word has many senses. WSD is to determine meanings of the ambiguous word based on its context, which is widely applied to natural language processing tasks. There are many ambiguous words in Chinese vocabulary. For example, Chinese word 'ben' has 3 semantic categories including 'book,' 'capital,' and 'foundation.' We need determine meanings of 'ben' based on its context. Now, scholars at home and abroad are studying WSD. WSD methods can be divided into 3 categories: supervised methods, unsupervised ones, and semisupervised ones.

In the supervised WSD method, annotated corpus is used to train the WSD classifier. The optimized classifier is adopted to disambiguate test corpus [1]. In the unsupervised WSD method, corpus need not be annotated manually. Unlabeled corpus is analyzed to reveal its inherent nature and law. Unlabeled corpus is disambiguated based on the similarity between unlabeled instances [2]. In the semisupervised WSD method, annotated corpus is used to train classifier. At the same time, a large amount of unannotated corpus is used to expand training corpus for improving the performance of the WSD classifier [3].

Our work is different from the method proposed by Trask [4]. Trask gives the Sense2Vec model based on Word2Vec. The Sense2Vec model is used to select appropriate sense embeddings of context for WSD. At the same time, he clusters ambiguous words with supervised labels.

The novelty of our work is that words, parts of speech, and semantic categories from all left and right units around ambiguous word are used as discriminative features. The WSD graph is adopted to express discriminative features and sentence containing the ambiguous word. Word2Vec tool is used to vectorize discriminative features, and Doc2Vec tool is used to vectorize the sentence containing the ambiguous word. Word2Vec, PMI, and TF-IDF are applied to compute embeddings of their relationships between discriminative features and sentence. GCN and softmax function are applied to WSD based on the WSD graph.

Main contributions of this article are summarized as follows:

(1) Sentence containing ambiguous word, word, part of speech, and semantic category are viewed as discriminative features. Doc2Vec tool is used to generate the feature vector of the sentence. Word2Vec tool is adopted to generate feature vectors of word, part of speech, and semantic category.

(2) WSD graph is constructed, in which discriminative features are used as nodes. Edges between word and sentence, word, part of speech, and semantic category are, respectively, constructed. PMI, TF-IDF, and Word2Vec tool are used to compute edge weights.

(3) WSD graph is input into GCN to determine the semantic category of the ambiguous word.

This paper is organized as follows. Related work is reported in Section 2. WSD feature extraction is given in Section 3. GCN word sense disambiguation is described in Section 4. Experimental results are given and analyzed in Section 5. Conclusion is described in Section 6.

2. Related Work

WSD is divided into supervised methods, unsupervised ones, and semisupervised ones.

In supervised WSD methods, labeled data is used to train WSD classifier. Zhang et al. [5] proposed two supervised WSD models based on the bidirectional long short-term memory network and self-attention model in the biomedical field. Amancio et al. [6] solved the WSD problem by treating texts as complex networks, where the semantic category of the ambiguous word is distinguished upon characterizing its local structure. Pal et al. [7] extended the baseline strategy and gave an improved WSD supervised method to establish decision trees, support vector machines, artificial neural networks, and naive Bayes models. Silva and Amancio [8] applied the framework of complex networks to the problem of supervised classification in word disambiguation task. Tripodi and Pelillo [9] designed the WSD model based on evolutionary game theory, which determines the semantic category of the ambiguous word according to distribution information and semantic similarity. Correa et al. [10] computed the relevance of the bipartite network representing both feature words and ambiguous word to solve ambiguities in written texts. Kumar et al. [11] presented a supervised method to disambiguate the ambiguous word by predicting over a continuous sense embedding space, which generalizes over both seen and unseen senses. Bevilacqua and Navilgli [12] embedded information of the LKB graph into a supervised neural architecture for exploiting pre-trained synset embeddings to predict ones that are not in the training set.

In unsupervised WSD methods, unlabeled corpus is clustered to determine the semantic category of the ambiguous word. Alsaeedan et al. [13] proposed a hybrid WSD method that consists of self-adaptive genetic algorithm, max-min ant one, and ant colony one. Meng et al. [14] give a context2vec model with part of speech to differentiate meanings represented by one point in vector space. Li et al. [15] presented the WSD method based on polysemy vector representation. Statistical polysemy, the number of word senses, and K-means clustering algorithm are adopted to disambiguate the ambiguous word. Yuan et al. [16] transformed WSD into text topic classification problem. They designed an unsupervised WSD model based on LDA topic. Jain and Lobiyal [17] applied the graph to reveal implicit information that links words in a sentence for WSD. Zhong and Wang [18] used the multiple kernel learning approach to combine multiple feature channels, which learns different weights that reflect the different importance of feature channels for WSD. Blevins and Zettlemoyer [19] designed a bi-encoder to embed ambiguous word with its surrounding context and dictionary definition. Encoders are jointly optimized in the same representation space and the nearest sense is selected. Ruas et al. [20] proposed a WSD method which considers semantic effects of contexts to disambiguate and annotate ambiguous word by its specific sense. Yang et al. [21] used domain keywords and word vector from unlabeled data to build WSD classifier. The proposed method is adapted to WSD task of other domains when knowledge from different fields is integrated. Hung and Chen [22] gave 3 methods based on contexts of word-of-mouth documents to build SentiWordNet lexicons for WSD. Lu et al. [23] combined Chinese and English knowledge resources by mapping word senses to construct a graph. At the same time, a graph-based WSD method with multi-knowledge integration is presented.

In semisupervised WSD methods, annotated corpus and a large amount of unannotated one are used to train the WSD classifier. Jain et al. [24] gave a semisupervised algorithm for constructing WordNet graphs, in which clue words are used. Saqib et al. [25] designed a framework consisting of buzz words and query words to use WordNet for detecting target words. Buzz words are defined as a ‘bag-of-words’ using POS, and query words have multiple meanings. Zhu [26] presented a semisupervised WSD method based on von Neumann kernel. Semantic similarities between terms are determined with both labeled and unlabeled data by means of a diffusion process on a graph defined by lexicon and co-occurrence. von Neumann kernel is constructed based on semantic similarity. Cardellino and Alonso Alemany [27] proposed a disjoint semisupervised learning method, in which an unsupervised model is trained on unlabeled data, and its results are used by a supervised classifier. Janz and Piasecki [28] combined plWordNet and semantic links extracted from a large valency lexicon, usage examples, Wikipedia articles, and SUMO ontology in a PageRank-based WSD algorithm. Mahmoodv and Hourali [29] used the machine learning algorithm with minimal supervision to disambiguate word senses based on features of target word and collaborative learning method. Başkaya and Jurgens [30] gave a semisupervised WSD method that combines a small amount of annotated data with information from word sense induction. Navigli and Velardi [31] created structural specifications of possible senses for each word in context and selected the best hypothesis with G grammar. Khapra et al. [32] adopted bilingual bootstrapping strategy for WSD, in which a model trained by annotated data is applied to annotate untagged data and vice versa using parameter projection. Akkaya et al. [33] used clustering and labeling strategy to generate labeled data for subjectivity WSD semiautomatically. Faralli and Navigli [34] designed a minimally-supervised framework for performing domain-driven WSD.

These 3 methods have their own shortcomings. Although the supervised WSD method can achieve the better performance, it needs a lot of annotated corpus. It is time-consuming and laborious. At the same time, the performance of WSD relies on machine learning algorithms. The unsupervised WSD method does not label corpus manually. But, disambiguation accuracy is not high. The semisupervised WSD method requires a small amount of annotated corpus and a large amount of unannotated one. But, it makes the WSD model worse and affects the coverage of ambiguous words to use unlabeled corpus to fit the model. GCN can capture global information of the graph and represent features of nodes better. Convolutional kernels of GCN act on all nodes of the whole graph, and weight parameters are shared. This reduces parameters of a single-layer network and effectively avoids the overfitting problem. Therefore, this paper proposes a WSD method based on GCN. Sentence, word, part of speech, and semantic category are viewed as nodes, and their relationships are used as edges. The WSD graph is constructed. GCN is adopted to extract effective features from the WSD graph and apply linguistic knowledge from corpus better to WSD.

3. WSD Feature Extraction

Firstly, the Chinese sentence including the ambiguous word is segmented into words. Secondly, the Chinese word is labeled with part of speech. Thirdly, the Chinese word is annotated with the semantic category. Here, word, part of speech, and semantic category are extracted from contexts of the ambiguous word as discriminative features. For Chinese sentence containing ambiguous word ‘biaomian,’ the process of extracting discriminative features is shown in Figure 1.

Here, w is a word, p denotes part of speech, s expresses the semantic category, and d is the Chinese sentence containing the ambiguous word.

Chinese sentence: Zhe yang ke qu chu shu cai biao mian de can liu nong yao

Word segmentation: Zheyang ke quchu shucai biaomian de canliu nongyao

Part of speech: Zheyang/*r* ke/*v* quchu/*v* shucai/*n* biaomian/*n* de/*u* canliu/*vn* nongyao/*n*

Semantic tagging: Zheyang/*r*/Ka34 ke/*v*/Ka01 quchu/*v*/Hg18 shucai/*n*/Bh06 biaomian/*n*/Bc02 de/*u*/Bo29 canliu/*vn*/Jd01 nongyao/*n*/Br13

Word2Vec tool is used to vectorize word, part of speech, and semantic category, respectively. Word2Vec(x) represents the vectorization result of x:

$$\begin{aligned}
v_{w1} &= \text{Word2Vec}(\text{Zheyang}), v_{w2} = \text{Word2Vec}(\text{ke}), \ldots, v_{w7} = \text{Word2Vec}(\text{nongyao}),\\
v_{p1} &= \text{Word2Vec}(r), v_{p2} = \text{Word2Vec}(v), \ldots, v_{p7} = \text{Word2Vec}(n),\\
v_{s1} &= \text{Word2Vec}(\text{Ka34}), v_{s2} = \text{Word2Vec}(\text{Ka01}), \ldots, v_{s7} = \text{Word2Vec}(\text{Br13}).
\end{aligned} \tag{1}$$

Doc2Vec tool is adopted to vectorize the sentence. Doc2Vec(x) denotes the vectorization result of x:

$$v_{d1} = \text{Doc2Vec}(\text{Zheyang Ke quchu shucai de canliu nongyao}), \tag{2}$$

where d_1 = ‘Zheyang ke quchu shucai de canliu nongyao.’

For the above sentence, 21 discriminative features are extracted including $w_1 = \text{Zheyang}$, $p_1 = r$, $s_1 = \text{Ka34}$, $w_2 = \text{ke}$, $p_2 = v$, $s_2 = \text{Ka01}$, $w_3 = \text{quchu}$, $p_3 = v$, $s_3 = \text{Hg18}$, $w_4 = \text{shucai}$, $p_4 = n$, $s_4 = \text{Bh06}$, $w_5 = \text{de}$, $p_5 = u$, $s_5 = \text{Bo29}$, $w_6 = \text{canliu}$, $p_6 = vn$, $s_6 = \text{Jd01}$, $w_7 = \text{nongyao}$, $p_7 = n$, and $s_7 = \text{Br13}$.

4. WSD Based on Graph Convolutional Network

GCN is a multilayer neural network that operates directly on a graph and induces the embedding vector of a node based on its neighbor ones. An undirected graph $G = (V, E)$ is defined, where $V(|V| = N)$ is a set of nodes and E is a set of edges. Assuming that each node is connected with itself, then any v has $(v, v) \in E$, $v \in V$.

The WSD graph of the corpus is constructed, whose nodes are sentence, word, part of speech, and semantic category. At the same time, the WSD graph contains embeddings of nodes and edge weights. It is important to construct edges, respectively, between word and sentence, word, part of speech, and semantic category. The set of sentence nodes is $U_D = \{D_1, D_2, \ldots\}$, and d_i is sentence in node D_i $(i = 1, 2, \ldots)$. The set of word nodes is $U_W = \{W_1, W_2, \ldots\}$, and w_i is word in node W_i $(i = 1, 2, \ldots)$. The set of part of speech nodes is $U_P = \{P_1, P_2, \ldots\}$, and p_i is part of speech in node P_i $(i = 1, 2, \ldots)$. The set of semantic nodes is $U_S = \{S_1, S_2, \ldots\}$, and s_i is the semantic category in node S_i $(i = 1, 2, \ldots)$. $V = U_D \cup U_W \cup U_P \cup U_S$. There are N nodes including D, W, P, and S in the WSD graph. Adjacency matrix $A \in R^{N\times N}$ is constructed based on V. Each node has a M-dimensional feature vector which is Word2Vec or Doc2Vec. N feature vectors form feature matrix $X \in R^{N\times M}$. Adjacency matrix A and feature matrix X are input into GCN. Then, the softmax function is adopted to determine the semantic category of the ambiguous word. GCN word sense disambiguation is shown in Figure 2.

There are t semantic categories $s_1, s_2, \ldots, s_t$ for the ambiguous word w, as shown in Figure 2. The ellipse represents node, and the line denotes the edge between two nodes.

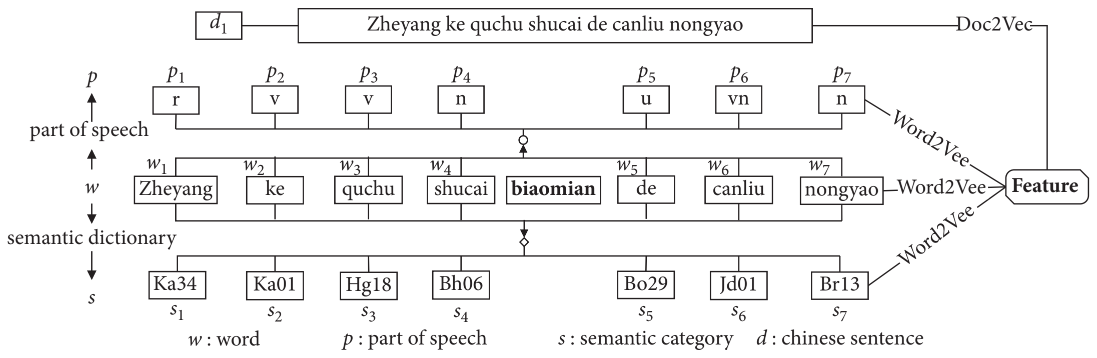

FIGURE 1: Feature extraction.

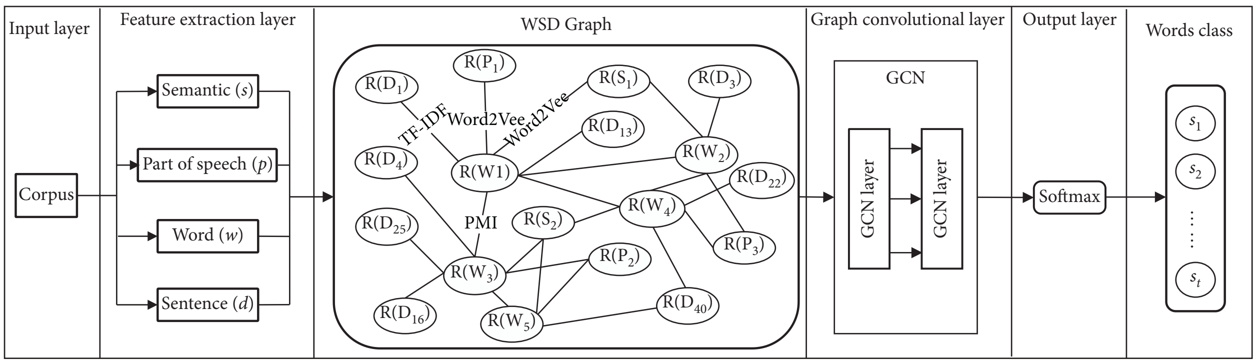

FIGURE 2: GCN word sense disambiguation.

Here, 'D' represents the sentence node, 'W' denotes the word node, 'P' represents the part of speech node, and 'S' denotes the semantic node. The number is used to distinguish different sentences, words, parts of speech, and semantic categories. $R(X)$ represents the embedding representation of X. Here, X can be sentence, word, part of speech, and semantic category. The edge between $R(D_1)$ and $R(W_1)$ denotes the relationship between D_1 and W_1, whose value is TF-IDF(d_1, w_1). The edge between $R(W_1)$ and $R(W_2)$ represents the relationship between W_1 and W_2, whose value is PMI (w_1, w_2). The edge between $R(W_1)$ and $R(P_1)$ denotes the relationship between W_1 and P_1, whose value is Word2Vec (p_1). The edge between $R(W_1)$ and $R(S_1)$ represents the relationship between W_1 and S_1, whose value is Word2Vec(s_1). The output of GCN is input into the softmax function to determine the semantic category of w.

Discriminative features are extracted from lexical units in the sentence including ambiguous word w. For the above example, d_1 represents the sentence. Doc2Vec tool is used to vectorize d_1 as v_{d1}. Word2Vec tool is adopted to vectorize word, part of speech, and semantic category, respectively, as v_w, v_p, and v_s. The same features are viewed as a node in the WSD graph. There are 20 nodes for the WSD graph of the above sentence. 200-dimensional feature vector is generated for each feature. Feature matrix $X \in R^{20\times 200}$ is constructed by 20 feature vectors. The process of constructing feature matrix X is shown in Figure 3.

We use PMI to calculate edge weight between two word nodes. A fixed-size sliding window is adopted to collect co-occurrence statistics. The sliding process of the window is shown in Figure 4.

In Figure 4, w represents the word and the number denotes location. Suppose there are 7 words in a sentence including $w_1, w_2, \ldots, w_7$. If the size of the sliding window is set to 3, the window contains 3 words. The window slides from left to right. When the window slides each time, a new word will be included and the first word of the window will be discarded. The sliding process does not stop until the rightest word of the window is the last word of the sentence.

PMI is adopted to calculate the weight between two words. PMI value of word pair w_i, w_j is defined as

$$\begin{aligned} \mathrm{PMI}\left(w_i, w_j\right) &= \log \frac{p\left(w_i, w_j\right)}{p\left(w_i\right)p\left(w_j\right)}, \\ p\left(w_i, w_j\right) &= \frac{\#\left(w_i, w_j\right)}{\#}, \\ p\left(w_i\right) &= \frac{\#\left(w_i\right)}{\#}, \end{aligned} \tag{3}$$

where $\#(w_i)$ is the number of sliding windows containing w_i, $\#(w_i, w_j)$ is the number of sliding windows containing w_i and w_j, and $\#$ is the number of sliding windows. When the PMI value is positive, semantic relevance between w_i and w_j is high. Otherwise, their semantic relevance is low. An edge is added between word nodes W_i and W_j when PMI (w_i, w_j) is positive.

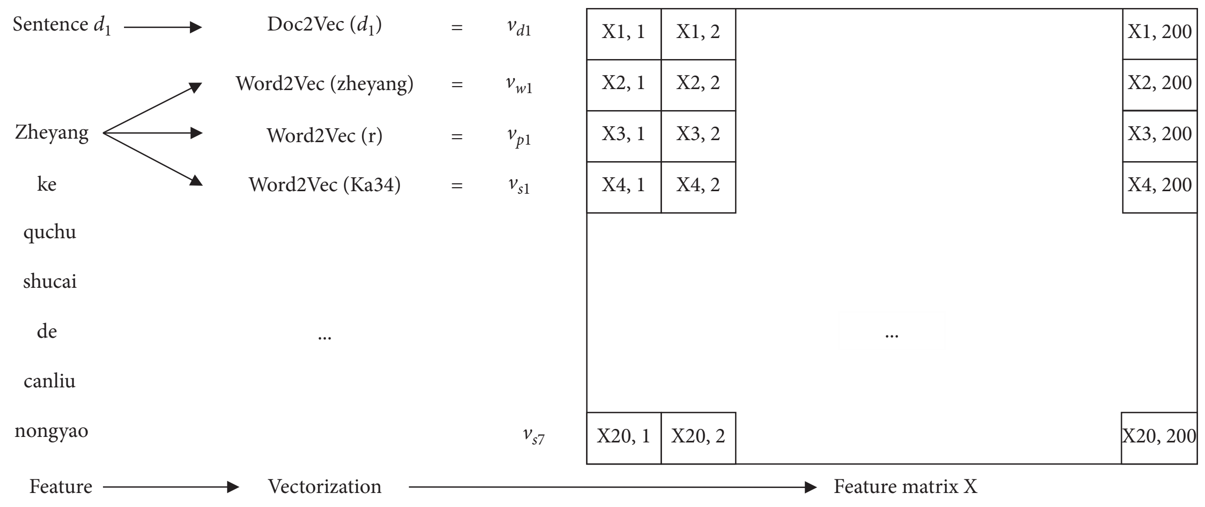

Figure 3: The process of constructing the feature matrix.

The process of constructing the WSD graph is shown as follows:

(1) Use Doc2Vec tool to vectorize the sentence as embedding of node D. Use Word2Vec tool to vectorize word, part of speech, and semantic category, respectively, as embeddings of node W, node P, and node S.

(2) When PMI (w_i, w_j) is positive, an edge is added between nodes W_i and W_j whose weight is PMI (w_i, w_j).

(3) Use Word2Vec tool to vectorize part of speech p_j in node P_j as edge weight between nodes W_i and P_j. Use Word2Vec tool to vectorize semantic category s_j in node S_j as edge weight between nodes W_i and S_j.

(4) Use TF-IDF to vectorize sentence d_i in node D_i and word w_j in node W_j, where TF is the frequency that word w_j occurs in d_i and IDF is logarithmic inverse fraction of the number of sentences containing word w_j.

Edge weight between nodes i and j can be given by

$$A_{ij} = \begin{cases} \mathrm{PMI}(w_i, w_j), & \text{Elements in node } i \text{ and } j \text{ are } w_i, w_j \text{ and } \mathrm{PMI}(w_i, w_j) > 0, \\ \mathrm{TF-IDF}(d_i, w_j), & \text{Elements in node } i \text{ and } j \text{ are } d_i, w_j, \\ \mathrm{Word2Vec}(p_j), & \text{Elements in node } i \text{ and } j \text{ are } w_i, p_j, \\ \mathrm{Word2Vec}(s_j), & \text{Elements in node } i \text{ and } j \text{ are } w_i, s_j, \\ 1, & i = j, \\ 0, & \text{otherwise}. \end{cases} \quad (4)$$

One layer GCN can obtain information of its neighbor nodes through convolution operations. When multiple GCN layers are stacked, information about larger neighborhoods are integrated. Feature matrix $L^{(1)} \in R^{N\times K}$ is defined as

$$L^{(1)} = \rho(\tilde{A}XW_0), \quad (5)$$

where ρ is the activation function and $W_0 \in R^{M\times K}$ is the weight matrix. Normalized symmetric adjacency matrix $\tilde{A}$ can be given by

$$\tilde{A} = D^{-(1/2)}AD^{-(1/2)}. \quad (6)$$

If $\tilde{A}$ is the ordinary adjacency matrix, it cannot consider the influence of a node on itself. At the same time, it cannot consider that a node with more neighbors has greater influence on WSD.

GCN aggregates high-order neighborhood information by stacking multiple convolutional layers, which makes a node have more information using the following formula:

$$L^{(j+1)} = \rho(\tilde{A}X^{(j)}W_j), \quad (7)$$

where j is the layer number of GCN and $L^{(0)} = X$.

Two-layer GCN is used here. Adjacency matrix A and feature matrix X are input into GCN. Then, the softmax function is adopted to compute probabilities of ambiguous word w under semantic categories, which are defined as

$$Z = \mathrm{soft\,max}(\tilde{A}\mathrm{ReLU}(\tilde{A}XW_0)W_1), \quad (8)$$

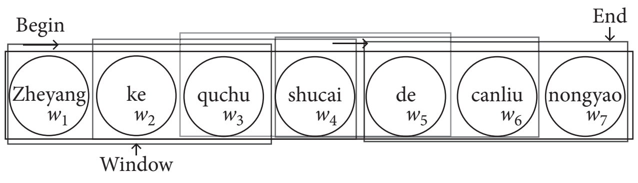

FIGURE 4: The sliding process of the window.

where ReLU is the activation function, W_0 is the weight matrix of the input layer, W_1 is the weight matrix of the output layer, $\tilde{A}XW_0$ is feature embeddings of the input layer, and $\mathrm{ReLU}(\tilde{A}XW_0)W_1$ is feature embeddings of the output layer.

Probability that ambiguous word w belongs to semantic category s_i can be given by

$$p(s_i|w) = \frac{e^{s_i}}{\sum_{i=1}^{t} e^{s_i}}, \tag{9}$$

where $U = \{(d_1, l_1), (d_2, l_2), \ldots, (d_n, l_n)\}$ and $l_i \in \{s_1, s_2, \ldots, s_t\}$ is the semantic category of sentence d_i ($i = 1, 2, \ldots, n$). Loss function L is defined as cross-entropy loss error of all annotated instances, which is defined as

$$L = -\sum_{i=1}^{n}\sum_{j=1}^{t} l_i \ln Z_{ij}, \tag{10}$$

where Z_{ij} is the predicted probability that d_i belongs to semantic category s_j, as shown in formula (8).

Parameters W_0 and W_1 of formula (8) can be trained by the gradient descent method based on U.

Semantic category s of ambiguous word w is determined using the following formula:

$$s = \underset{i=1,2,\ldots,t}{\arg\max}\, p(s_i|w). \tag{11}$$

5. Experimental Results and Analysis

Training corpus of SemEval-2007: Task #5 is used to optimize the proposed WSD classifier. Test corpus of SemEval-2007: Task #5 is adopted to testify the performance of the optimized classifier. 28 ambiguous words are selected from SemEval-2007: Task #5.16 ambiguous words have two semantic categories. They are, respectively, 'biaomian,' 'cai,' 'danwei,' 'dongyao,' 'ernv,' 'jingtou,' 'kaitong,' 'qixi,' 'qixiang,' 'shi,' 'tuifan,' 'wang,' 'yanguang,' 'zhenjing,' and 'zhongyi.' 10 ambiguous words with 3 semantic categories are selected including 'ben,' 'bu,' 'chengli,' 'duiwu,' 'gan,' 'qizhi,' 'rizi,' 'tiandi,' 'tiao,' and 'changcheng.' 3 ambiguous words have 4 semantic categories. They are, respectively, 'chi,' 'dong,' and 'jiao.'

In order to testify the proposed method, we compare the influence of LSTM, CNN, CNN + LSTM, CNN + Bi-LSTM, and GCN on WSD. Four groups of experiments are conducted. The first and second groups of experiments are performed to compare the performance of LSTM-based, CNN-based, CNN + LSTM-based, CNN + Bi-LSTM-based WSD methods, and the proposed one under different discriminative features. The third group of experiments are conducted to testify the influence of window size on WSD. The fourth group of experiments are performed to testify the influence of the convolutional layer number on WSD.

Average accuracy is used to evaluate the performance of the WSD classifier, which is defined as

$$\begin{aligned} p_i &= \frac{m_i}{n_i}, \\ p_{\mathrm{avg}} &= \frac{\sum_{i=1}^{N} p_i}{N}, \end{aligned} \tag{12}$$

where N is the number of all ambiguous words, m_i is the number of test sentences correctly classified for the ith ambiguous word, n_i is the number of all test sentences containing the ith ambiguous word, p_i is disambiguation accuracy of the ith ambiguous word, and p_{avg} is average accuracy.

In the first two groups of experiments, we compare some WSD methods based on deep neural networks and proposed one. In the first group of experiments, words are extracted as discriminative features from two left and right units around the ambiguous word. LSTM, CNN, CNN + LSTM, and CNN + Bi-LSTM are, respectively, adopted to determine its semantic category. In GCN(1), words are extracted as discriminative features from two left and right units around the ambiguous word. 4 words compose d. Discriminative features and d are used to construct the WSD graph. In GCN(2), words are extracted as discriminative features from all left and right units of the ambiguous word. These words compose d. Discriminative features and d are used to construct the WSD graph. Training corpus is used to optimize the WSD model, and test corpus is adopted to testify accuracy of the WSD model, as shown in Table 1.

From Table 1, it can be seen that WSD performance of GCN is higher than those of other models. Average accuracy of GCN(2) is superior to that of GCN(1). This is because that all left and right words around ambiguous word are used in GCN(2). But, two left and right words around the ambiguous word are only used in GCN(1). GCN(2) considers more contexts than GCN(1). So, GCN(2) is better than GCN(1) on WSD performance. Average accuracy of LSTM is slightly lower than that of CNN, which shows that CNN can extract disambiguation features better. CNN and LSTM are inferior to CNN + LSTM on WSD performance. The reason is that CNN + LSTM can extract more effective features than CNN and LSTM. Average accuracy of CNN + LSTM is the same with that of CNN + Bi-LSTM. It indicates that they have the same ability of feature extraction when they are used, respectively, to extract disambiguation features from two left and right words around ambiguous words.

In the second group of experiments, words, parts of speech, and semantic categories are extracted as discriminative features from two left and right units around the ambiguous word. LSTM, CNN, CNN + LSTM, and CNN + Bi-LSTM are adopted to determine its semantic category. In CNN(1), frequencies of discriminative features

Table 1: Disambiguation accuracies in the first group of experiments.

Ambiguous words	LSTM	CNN	CNN + LSTM	CNN + Bi-LSTM	GCN(1)	GCN(2)
Biaomian(2)	0.611	0.611	0.611	0.611	0.833	0.889
Cai(2)	0.524	0.550	0.476	0.524	0.737	0.737
Danwei(2)	0.632	0.632	0.632	0.632	0.882	0.706
Dongyao(2)	0.625	0.625	0.625	0.625	0.563	0.688
Ernv(2)	0.545	0.524	0.545	0.545	0.850	1.000
Jingtou(2)	0.533	0.533	0.533	0.533	0.568	0.600
Kaitong(2)	0.500	0.526	0.500	0.500	0.500	0.700
Qixi(2)	0.286	0.714	0.714	0.714	0.714	1.000
Qixiang(2)	0.647	0.647	0.647	0.647	0.688	0.938
Shi(2)	0.625	0.625	0.625	0.625	0.813	0.813
Tuifan(2)	0.400	0.600	0.600	0.600	0.600	0.800
Wang(2)	0.769	0.769	0.769	0.769	0.769	0.769
Yanguang(2)	0.714	0.714	0.714	0.714	0.571	0.714
Zhenjing(2)	0.714	0.714	0.714	0.714	0.857	0.786
Zhongyi(2)	0.480	0.458	0.480	0.480	0.813	0.688
Ben(3)	0.429	0.370	0.429	0.429	0.760	0.760
Bu(3)	0.500	0.522	0.500	0.500	0.750	0.850
Chengli(3)	0.357	0.370	0.357	0.357	0.357	0.393
Duiwu(3)	0.458	0.478	0.458	0.417	0.546	0.636
Gan(3)	0.450	0.421	0.450	0.450	0.889	0.611
Qizhi(3)	0.556	0.556	0.556	0.556	0.667	0.722
Rizi(3)	0.333	0.344	0.333	0.333	0.406	0.594
Tiandi(3)	0.400	0.375	0.400	0.400	0.800	0.720
Tiao(3)	0.438	0.438	0.438	0.438	0.643	0.714
Changcheng(3)	0.577	0.600	0.577	0.577	0.524	0.571
Chi(4)	0.500	0.259	0.500	0.500	0.478	0.565
Dong(4)	0.542	0.528	0.542	0.542	0.500	0.500
Jiao(4)	0.250	0.263	0.250	0.250	0.359	0.410
Average accuracy	0.514	0.527	0.535	0.535	0.659	0.710

are used and CNN is used to determine its semantic category [35]. In GCN(3), words, parts of speech, and semantic categories are extracted as discriminative features from two left and right units of the ambiguous word. 4 words compose *d*. Discriminative features and *d* are used to construct the WSD graph. In GCN(4), words, parts of speech, and semantic categories are extracted as discriminative features from all left and right units around the ambiguous word. These words compose *d*. Discriminative features and *d* are used to construct the WSD graph. Training corpus is used to optimize the WSD model, and test corpus is adopted to testify accuracy of the WSD model, as shown in Table 2.

From Table 2, it can be seen that WSD performance of LSTM, CNN, CNN + LSTM, CNN + Bi-LSTM, and GCN(4) is improved compared with the corresponding model in the first group of experiments. This is because that words, parts of speech, and semantic categories are used as discriminative features. It shows that part of speech and semantic category provide more discriminative information for WSD. Average accuracy of CNN is better than that of LSTM. It indicates that CNN can extract disambiguation features better from words, parts of speech, and semantic categories. CNN + LSTM is superior to CNN and LSTM on WSD performance. The reason is that CNN + LSTM can extract more effective features than CNN and LSTM. Average accuracy of CNN + LSTM is lower than that of CNN + Bi-LSTM. It indicates that CNN + Bi-LSTM has the better ability of feature extraction when they are used, respectively, to extract disambiguation features from two left and right words, parts of speech, and semantic categories around ambiguous words. Average accuracy of CNN(1) is higher than that of CNN. It shows that average accuracy of CNN is improved when frequencies are vectorized as disambiguation features. Average accuracy of GCN(4) is higher than that of GCN(3). This is because that all left and right words, parts of speech, and semantic categories around ambiguous word are used in GCN(4). But, two left and right words, parts of speech, and semantic categories around the ambiguous word are only used in GCN(3). GCN(4) considers more contexts than GCN(3). So, GCN(4) is better than GCN(3) on WSD performance. But, average accuracy of GCN(3) is lower than GCN(1). It shows that it influences information transfer between nodes to introduce more discriminative features into GCN when there are few nodes.

Ambiguous words of GCN(2) and GCN(4) in Tables 2 and 3 are classified according to the category number. Average accuracy of ambiguous words with the same category number is calculated, as shown in Figure 5.

From Figure 5, it can be seen that average accuracy of ambiguous words with the same category number decreases when the category number increases. This is because that the predicted results have more possibilities with the category number increasing. It makes error rate of the WSD classifier higher. Average accuracy of GCN(4) is better than that of GCN(2) for 2 categories, 3 ones and 4 ones. This is because

Table 2: Disambiguation accuracies in the second group of experiments.

Ambiguous word	LSTM	CNN	CNN + LSTM	CNN + Bi-LSTM	CNN(1)	GCN(3)	GCN(4)
Biaomian(2)	0.611	0.611	0.611	0.611	0.823	0.800	0.889
Cai(2)	0.474	0.474	0.526	0.526	0.722	0.833	0.790
Danwei(2)	0.632	0.632	0.632	0.632	0.823	0.800	0.765
Dongyao(2)	0.625	0.625	0.625	0.625	0.937	0.368	0.625
Ernv(2)	0.455	0.545	0.545	0.545	0.949	0.536	1.000
Jingtou(2)	0.533	0.533	0.533	0.533	0.533	0.348	0.600
Kaitong(2)	0.500	0.500	0.500	0.500	0.850	0.824	0.650
Qixi(2)	0.714	0.714	0.714	0.714	0.642	0.250	1.000
Qixiang(2)	0.647	0.647	0.647	0.647	0.875	0.563	0.938
Shi(2)	0.625	0.625	0.625	0.625	0.812	0.546	0.813
Tuifan(2)	0.600	0.600	0.600	0.600	0.600	0.250	0.900
Wang(2)	0.769	0.769	0.769	0.769	0.923	0.222	0.692
Yanguang(2)	0.714	0.714	0.714	0.714	0.714	0.641	0.643
Zhenjing(2)	0.714	0.714	0.714	0.714	0.928	0.533	0.786
Zhongyi(2)	0.526	0.526	0.526	0.526	0.875	0.500	0.750
Ben(3)	0.370	0.444	0.444	0.444	0.989	0.778	0.760
Bu(3)	0.478	0.478	0.478	0.478	0.649	0.214	0.850
Chengli(3)	0.357	0.357	0.357	0.357	0.666	0.250	0.643
Duiwu(3)	0.458	0.458	0.458	0.458	0.590	0.344	0.636
Gan(3)	0.474	0.474	0.474	0.474	0.555	0.375	0.611
Qizhi(3)	0.556	0.556	0.556	0.556	0.722	0.080	0.944
Rizi(3)	0.333	0.333	0.333	0.364	0.500	0.214	0.594
Tiandi(3)	0.400	0.400	0.400	0.400	0.639	0.500	0.880
Tiao(3)	0.467	0.467	0.467	0.467	0.571	0.231	0.571
Changcheng(3)	0.577	0.577	0.577	0.577	0.714	0.714	0.476
Chi(4)	0.500	0.500	0.500	0.500	0.521	0.524	0.783
Dong(4)	0.524	0.524	0.524	0.524	0.500	0.143	0.550
Jiao(4)	0.250	0.250	0.250	0.275	0.700	0.125	0.410
Average accuracy	0.532	0.537	0.539	0.541	0.726	0.447	0.734

that words are only adopted as discriminative features in GCN(2). Words, parts of speech, and semantic categories are used as discriminative features in GCN(4). Compared with GCN(2) and GCN(4), average accuracy growth of ambiguous words with more categories is bigger than that of ambiguous words with less ones. This shows that, for the ambiguous word with more categories, more discriminative information will improve classification accuracy.

In the third group of experiments, words, parts of speech, and semantic categories are extracted as discriminative features from all left and right units around the ambiguous word. These words compose *d*. Discriminative features and *d* are used to construct the WSD graph, and GCN is adopted to determine the semantic category of the ambiguous word. The PMI value is applied to construct edges between word nodes in the WSD graph and compute their weights. PMI value is relevant to the size of the sliding window. By setting different window sizes, the influence of edge weight between two words on WSD is compared. Window size is, respectively, set to 5, 8, 10, 12, 15, 20, and the length of *d* #. Training corpus is used to optimize GCN in the third group of experiments. Test corpus is adopted to testify accuracy of the WSD model, as shown in Table 3.

From Table 3, it can be seen that average accuracy of WSD is the highest when window size is 10. Through the analysis of experimental results, it can be concluded that if window size is too small, the information between some word nodes is not aggregated well. Otherwise, there will be an edge between two irrelevant word nodes. When the window is expanded, accuracies of some ambiguous words increase and accuracies of some ambiguous words become worse. This is caused by the inconsistency of the word number in sentences. When window size is 10, the sliding window can cover the sentence length in corpus equally and a better WSD effect can be gotten. In order to observe the influence of window size more intuitively, a line chart of average accuracies under different window sizes is drawn, as shown in Figure 6.

In the fourth group of experiments, words, parts of speech, and semantic categories are extracted as discriminative features from all left and right units around the ambiguous word. These words compose *d*. Discriminative features and *d* are used to construct the WSD graph, and GCN is adopted to determine the semantic category of the ambiguous word. Window size is set to 10, and the convolutional layer number of GCN is set to different values. Training corpus is used to optimize GCN. Test corpus is adopted to testify accuracy of the WSD classifier, as shown in Table 4.

From Table 4, it can be seen that the effect of the WSD classifier is not good under 1 convolutional layer. This is because that the information between nodes cannot be fused well. In order to make a node have more extensive information, multiple convolutions are needed. The performance of WSD is the highest under 2 convolutional layer. When the layer number is 3 and 4, average accuracies of the WSD classifier decrease. This is because that, after the layer

Table 3: Disambiguation accuracies in the third group of experiments.

Ambiguous word	#	5	8	10	12	15	20
Biaomian(2)	0.8889	0.7222	0.9444	0.9444	0.8889	0.8889	0.8889
Cai(2)	0.7895	0.8421	0.8421	0.8421	0.8421	0.7895	0.7895
Danwei(2)	0.7647	0.7647	0.7647	0.8235	0.7059	0.7647	0.7647
Dongyao(2)	0.6250	0.5000	0.5625	0.6250	0.6250	0.6250	0.6250
Ernv(2)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Jingtou(2)	0.6000	0.4000	0.4667	0.5333	0.5333	0.5333	0.5333
Kaitong(2)	0.6500	0.6000	0.6500	0.6500	0.6500	0.7000	0.6500
Qixi(2)	1.0000	0.7857	1.0000	1.0000	1.0000	1.0000	1.0000
Qixiang(2)	0.9375	0.8750	1.0000	1.0000	1.0000	1.0000	1.0000
Shi(2)	0.8125	0.8125	0.8750	0.8750	0.8125	0.8125	0.7500
Tuifan(2)	0.9000	0.9000	0.9000	0.9000	0.9000	0.9000	0.8000
Wang(2)	0.6923	0.6923	0.6923	0.6923	0.6923	0.6923	0.6154
Yanguang(2)	0.6429	0.7143	0.6429	0.7143	0.6429	0.5714	0.5714
Zhenjing(2)	0.7857	0.7857	0.7143	0.7857	0.7857	0.7857	0.7857
Zhongyi(2)	0.7500	0.6875	0.6875	0.7500	0.6250	0.6250	0.6250
Ben(3)	0.7600	0.7200	0.7200	0.7600	0.7200	0.7200	0.7200
Bu(3)	0.8500	0.7000	0.8000	0.8500	0.8000	0.8500	0.8500
Chengli(3)	0.6429	0.5714	0.5714	0.6071	0.5714	0.5714	0.6071
Duiwu(3)	0.6364	0.5455	0.5455	0.5909	0.5909	0.6364	0.6364
Gan(3)	0.6111	0.6667	0.6111	0.6667	0.6667	0.6111	0.6111
Qizhi(3)	0.9444	0.8889	0.8889	0.9444	0.9444	0.8889	0.9444
Rizi(3)	0.5938	0.6250	0.5938	0.5313	0.5625	0.5625	0.5938
Tiandi(3)	0.8800	0.7200	0.7600	0.7600	0.8000	0.8000	0.8000
Tiao(3)	0.5714	0.5000	0.5714	0.6429	0.5714	0.5714	0.5714
Changcheng(3)	0.4762	0.5714	0.4762	0.4762	0.5238	0.5238	0.4286
Chi(4)	0.7826	0.7391	0.6957	0.7826	0.7391	0.7391	0.7391
Dong(4)	0.5500	0.4500	0.4500	0.5500	0.5000	0.4500	0.4500
Jiao(4)	0.4103	0.4103	0.4103	0.4359	0.3846	0.3590	0.3846
Average accuracy	0.7339	0.6854	0.7085	0.7409	0.7171	0.7133	0.7048

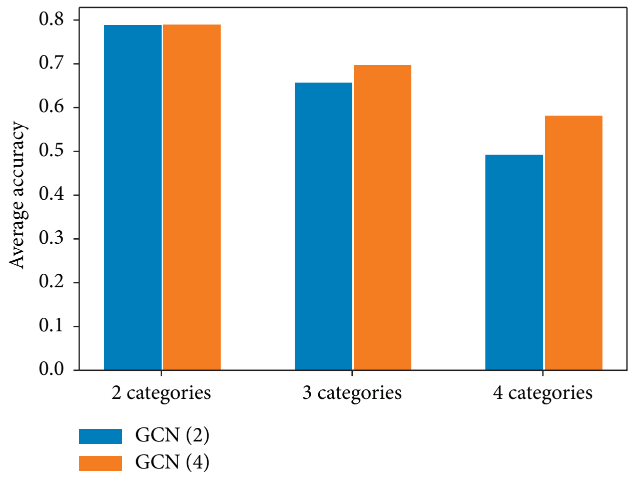

Figure 5: Average accuracy under different discriminative features and category number.

number increases, discriminative information in convolutional operations is more sufficient. The information in distant neighbors of a node will gradually gather. Information in irrelevant nodes converges, which leads to excessive information fusion and reduces the performance of the WSD classifier. Here, 'biaomian,' 'qixi,' 'zhongyi,' 'tiao,' 'changcheng,' and 'chi' are used as representative ambiguous words. Disambiguation accuracies under different layer numbers are shown in Figure 7.

From Figure 7, it can be seen that most ambiguous words have the highest accuracies when the layer number is 2. Accuracy of 'zhongyi' is the highest when the layer number is 3. When the layer number is 4, 'changcheng' has the highest accuracy. This shows that the effect of information fusion between nodes is the best when the layer number is high for some ambiguous words. Accuracies of some ambiguous words reduce because of excessive information fusion.

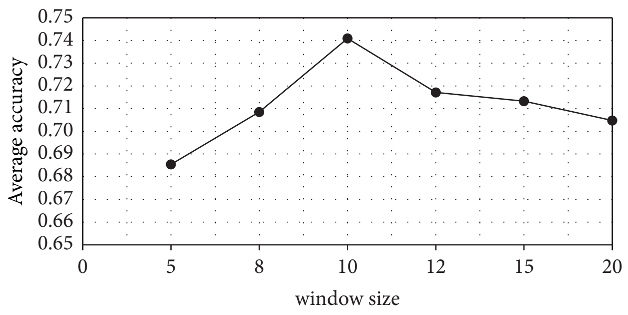

Figure 6: Average accuracies under different window sizes.

Table 4: Disambiguation accuracies in the fourth group of experiments.

Ambiguous word	1 layer	2 layers	3 layers	4 layers
Biaomian(2)	0.9444	0.9444	0.8889	0.7222
Cai(2)	0.6316	0.8421	0.5790	0.7368
Danwei(2)	0.7059	0.8235	0.6471	0.7059
Dongyao(2)	0.5625	0.6250	0.5000	0.5625
Ernv(2)	0.9500	1.0000	1.0000	1.0000
Jingtou(2)	0.4667	0.5333	0.4667	0.4667
Kaitong(2)	0.5000	0.6500	0.6000	0.6000
Qixi(2)	0.8571	1.0000	0.7143	0.7143
Qixiang(2)	0.8750	1.0000	0.8750	0.7500
Shi(2)	0.7500	0.8750	0.6875	0.6875
Tuifan(2)	0.8000	0.9000	0.8000	0.9000
Wang(2)	0.6154	0.6923	0.3846	0.7692
Yanguang(2)	0.6429	0.7143	0.6429	0.6429
Zhenjing(2)	0.7857	0.7857	0.7143	0.7143
Zhongyi(2)	0.4375	0.7500	0.8750	0.6875
Ben(3)	0.6400	0.7600	0.7200	0.6800
Bu(3)	0.7000	0.8500	0.7000	0.8000
Chengli(3)	0.5714	0.6071	0.6071	0.5000
Duiwu(3)	0.4546	0.5909	0.5000	0.5909
Gan(3)	0.5556	0.6667	0.5556	0.5556
Qizhi(3)	0.8333	0.9444	0.8333	0.7779
Rizi(3)	0.5313	0.5313	0.5000	0.4688
Tiandi(3)	0.6800	0.7600	0.8000	0.6400
Tiao(3)	0.3571	0.6429	0.4286	0.5000
Changcheng(3)	0.4286	0.4762	0.5238	0.5714
Chi(4)	0.7826	0.7826	0.6087	0.6087
Dong(4)	0.5000	0.5500	0.5000	0.3500
Jiao(4)	0.3590	0.4359	0.4872	0.4615
Average accuracy	0.6503	0.7409	0.6478	0.6487

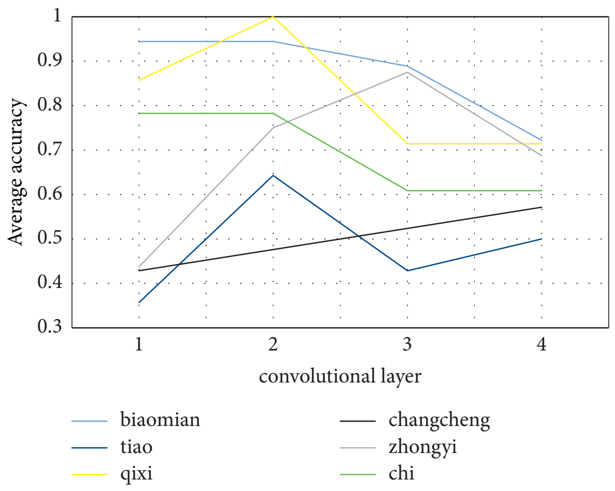

Figure 7: Accuracies under different layer numbers.

6. Conclusions

This paper proposes a WSD method based on GCN, in which words, parts of speech, and semantic categories from all left and right units around the ambiguous word are used as discriminative features. The WSD graph is constructed whose nodes are, respectively, words, parts of speech, semantic categories, and sentence. Edges are added, respectively, between word and sentence, word, part of speech, and semantic category. GCN is adopted to process the WSD graph, and the softmax function is used to determine the semantic category of the ambiguous word. We use the WSD graph to describe discriminative features, sentence, and their relationships, which can better express relationships between linguistic knowledge instead of serializing them. Experimental results show that GCN has better WSD performance than CNN, LSTM, CNN + LSTM, and CNN + Bi-LSTM and whether words are used as discriminative features or words, parts of speech, and semantic categories are used as discriminative features. GCN can make information transfer more fully between nodes of the WSD graph and extract disambiguation information effectively. It can obtain better WSD performance without too many convolutional layers, which greatly reduces the amount of calculation. In the future, we will integrate more language knowledge into the WSD graph and use the attention mechanism to further improve WSD performance.

Data Availability

SemEval-2007: Task#5 can be downloaded from http://ixa2.si.ehu.eus/clirwsd/.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. 61502124 and 60903082), Fundamental Research Fundation for Universities of Heilongjiang Province (Grant no. LGYC2018JC014), and Natural Science Foundation of Heilongjiang Province of China (Grant no. F2017014).

References

[1] A. Jimeno Yepes, “Word embeddings and recurrent neural networks based on long-short term memory nodes in supervised biomedical word sense disambiguation,” *Journal of Biomedical Informatics*, vol. 73, pp. 137–147, 2017.

[2] Y. Liu and H. Sun, “Word sense disambiguation for Chinese based on semantics calculation,” *Mathematical Problems in Engineering*, vol. 2015, Article ID 235096, 6 pages, 2015.

[3] J. Chen, L. Zhong, and C. Cai, “Using exponential kernel for semi-supervised word sense disambiguation,” *Journal of Computational and Theoretical Nanoscience*, vol. 13, no. 10, pp. 6929–6934, 2016.

[4] A. Trask, P. Michalak, and J. Liu, “Sense2vec-a fast and accurate method for word sense disambiguation in neural word embeddings,” 2015, http://arxiv.org/abs/1511.06388v1.

[5] C. Zhang, D. Biś, X. Liu, and Z. He, “Biomedical word sense disambiguation with bidirectional long short-term memory and attention-based neural networks,” *BMC Bioinformatics*, vol. 20, no. S16, p. 502, 2019.

[6] D. R. Amancio, O. N. Oliverira, and L. Costa, “Unveiling the relationship between complex networks metrics and word senses,” *Epl*, vol. 98, no. 1, 2012.

[7] A. R. Pal, D. Saha, N. S. Dash, S. K. Naskar, and A. Pal, “A novel approach to word sense disambiguation in Bengali language using supervised methodology,” *Sadhana*, vol. 44, no. 8, pp. 1–12, 2019.

[8] T. C. Silva and D. R. Amancio, “Word sense disambiguation via high order of learning in complex networks,” *Epl*, vol. 98, no. 5, pp. 1101–1104, 2012.

[9] R. Tripodi and M. Pelillo, “A game-theoretic approach to word sense disambiguation,” *Computational Linguistics*, vol. 43, no. 1, pp. 31–70, 2017.

[10] E. A. Corrêa, A. A. Lopes, and D. R. Amancio, “Word sense disambiguation: a complex network approach,” *Information Sciences*, vol. 442-443, pp. 103–113, 2018.

[11] S. Kumar, S. Jat, K. Saxena, and P. Talukdar, “Zero-shot word sense disambiguation using sense definition embeddings,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5670–5681, Florence, July 2019.

[12] M. Bevilacqua and R. Navigli, “Breaking through the 80% glass ceiling: raising the state of the art in word sense disambiguation by incorporating knowledge graph information,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2854–2864, July 2020.

[13] W. Alsaeedan, M. E. B. Menai, and S. Al-Ahmadi, “A hybrid genetic-ant colony optimization algorithm for the word sense disambiguation problem,” *Information Sciences*, vol. 417, pp. 20–38, 2017.

[14] Y. G. Meng, Q. L. Zhou, G. P. Zhang, and D. F. Cai, “Word sense disambiguation based on context similarity with POS/tagging,” *Journal of Chinese Information Processing*, vol. 32, no. 8, pp. 9–18, 2018.

[15] G. J. Li, Y. D. Zhao, and H. Q. Guo, “A word sense disambiguation method based on polysemy vector representation,” *Intelligent Computer and Applications*, vol. 8, no. 4, pp. 52–56, 2018.

[16] Y. Yuan, X. Li, and Y. T. Yang, “Unsupervised word sense disambiguation for Uyghur based on LDA topic model,” *Journal of Xiamen University(Natural Science)*, vol. 59, no. 2, pp. 198–205, 2020.

[17] G. Jain and D. K. Lobiyal, “Word sense disambiguation using implicit information,” *Natural Language Engineering*, vol. 26, no. 4, pp. 413–432, 2020.

[18] L. Y. Zhong and T. H. Wang, “Towards word sense disambiguation using multiple kernel support vector machine,” *International Journal of Innovative Computing, Information and Control*, vol. 16, no. 2, pp. 555–570, 2020.

[19] T. Blevins and L. Zettlemoyer, “Moving down the long tail of word sense disambiguation with gloss informed bi-encoder,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1006–1017, July 2020.

[20] T. Ruas, W. Grosky, and A. Aizawa, “Multi-sense embeddings through a word sense disambiguation process,” *Expert Systems with Applications*, vol. 136, pp. 288–303, 2019.

[21] A. Yang, S. J. Li, and Y. Li, “Word sense disambiguation based on domain knowledge and word vector model,” *Acta Scientiarum Nauralium Universitaties Pekinensis*, vol. 53, no. 2, pp. 204–210, 2016.

[22] C. Hung and S.-J. Chen, “Word sense disambiguation based sentiment lexicons for sentiment classification,” *Knowledge-Based Systems*, vol. 110, no. 1, pp. 224–232, 2016.

[23] W. Lu, F. Meng, S. Wang et al., “Graph-based Chinese word sense disambiguation with multi-knowledge integration,” *Computers, Materials & Continua*, vol. 61, no. 1, pp. 197–212, 2019.

[24] A. Jain, D. Kumar Tayal, and S. Vij, “A semi-supervised graph-based algorithm for word sense disambiguation,” *Global Journal of Enterprise Information System*, vol. 8, no. 2, pp. 13–19, 2017.

[25] S. M. Saqib, F. M. Kundi, and S. Ahmad, “Semi-supervised method for detection of ambiguous word and creation of sense: using wordNet,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 11, pp. 353–359, 2018.

[26] W. S. Zhu, “Semi-supervised word sense disambiguation using von neumann kernel,” *International Journal of Innovative Computing Information & Control*, vol. 13, no. 2, pp. 695–701, 2017.

[27] C. Cardellino and L. Alonso Alemany, “Exploring the impact of word embeddings for disjoint semisupervised Spanish verb sense disambiguation,” *Inteligencia Artificial*, vol. 21, no. 61, pp. 67–81, 2018.

[28] A. Janz and M. Piasecki, “A weakly supervised word sense disambiguation for Polish using rich lexical resources,” *Poznań Studies in Contemporary Linguistics*, vol. 55, no. 2, pp. 339–365, 2019.

[29] M. Mahmoodv and M. Hourali, “Semi-supervised approach for Persian word sense disambiguation,” in *Proceedings of the 7th International Conference on Computer and Knowledge Engineering*, pp. 104–110, Mashhad, October 2017.

[30] O. Başkaya and D. Jurgens, “Semi-supervised learning with induced word senses for state of the art word sense disambiguation,” *Journal of Artificial Intelligence Research*, vol. 55, no. 1, pp. 1025–1058, 2016.

[31] R. Navigli and P. Velardi, “Structural semantic interconnections: a knowledge-based approach to word sense disambiguation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 7, pp. 1075–1086, 2005.

[32] M. M. Khapra, S. Joshi, A. Chatterjee, and P. Bhattacharyya, “Together we can: bilingual bootstrapping for WSD,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pp. 561–569, Portland, June 2011.

[33] C. Akkaya, J. Wiebe, and R. Mihalcea, “Iterative constrained clustering for subjectivity word sense disambiguation,” in *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 269–278, Sweden, April 2014.

[34] S. Faralli and R. Navigli, “A new minimally-supervised framework for domain word sense disambiguation,” in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1411–1422, Korea, July 2012.

[35] C. X. Zhang, L. Y. Zhao, and X. Y. Gao, “Word sense disambiguation based on convolution neural network,” *Journal of Beijing University of Posts and Telecommunications*, vol. 42, no. 3, pp. 114–119, 2019.