DOI: 10.1051/itmconf/20171201027

Chinese Word Sense Disambiguation Using a LSTM

Xue-Ren SUN, Shao-He LV, Xiao-Dong WANG, Dong WANG

National Key Lab of Parallel and Distributed Computing,National University of Defense Technology,Changsha, China
snowman1003@qq.com,shaohelv@nudt.edu.cn, xdwang@nudt.edu.cn, dongwang@nudt.edu.cn

Abstract—Word sense disambiguation (WSD) is a challenging natural language processing (NLP) problem. We propose a new strategy for WSD, which at first replaces the interesting word in a sentence by the different synonyms corresponding to the different meanings, and then justify whether the transformed sentence is “legal”. A legal sentence is still legal after one or more word are replaced by other ones with the same meaning. A long short-term memory (LSTM) network-based model is proposed to perform the sentence/text classification. Furthermore, we build a Chinese WSD dataset based on HIT-CIR Tongyici Cilin (Extended) dataset. The model is evaluated on the new dataset and achieves better performance than the state-of-the-art.

1 Introduction

Word sense disambiguation is a challenging problem in natural language processing. It’s a basic problem in semantics and a supportive technology for many semantic problem, such as sentiment analysis, semantic analysis and so on.

It is known that, a word may have multiple meanings, no matter in Chinese or English. For example, a Chinese word “zu” have two common meanings, which are “si wang” and “shi bing”. In English, “death” and “soldier” are the meanings. However, in a specific context, the meaning of a word should be doubtless. For instance, there is a Chinese sentence:

Zhu Geliang sheng yu 181 nian, zu yu 234 nian.

In this context, the Chinese word “zu” has no ambiguity, it definitely means “death”. The word sense disambiguation problem aims to find the proper meaning of a word in a specific context automatically. Specific to this example, we should find the meaning “death” automatically based on the context.

In this paper, we use a new strategy to convert the word sense disambiguation problem into a text classification problem. The strategy replaces a word in a context with its different synonyms corresponding to its different meaning. For example, the word “zu” in the sentence above can be replaced with “si wang” and “shibing” respectively. As a result, the sentence above are converted into two sentences. We consider the sentence replaced with “si wang” correct, and the other one wrong. Using this rule, we can convert sentences with words being disambiguated into two sets: the positive set and the negative set. What we should do now is to train a model to have the ability to distinguish whether a sentence is correct or wrong.

To this end, we propose a novel classifier model based on long short term memory(LSTM) network [1]. The model consists of a LSTM and a multilayer perceptron(MLP). We adopt an LSTM network to model the long-range semantics of a sentence and an MLP to perform the classification.

As a supervised machine learning model, training is an essential step. And the training step needs large data to complete the work. So far, there has not been a Chinese WSD dataset large enough. To satisfy the need of the training, we build a Chinese WSD dataset based on HIT-CIR Tongyici Cilin (Extended) [2] dataset.

The HIT-CIR Tongyici Cilin (Extended) dataset encodes 77343 words with a perfect rule based on synonyms. Moreover, a corpus from People's Daily is also labeled by the same rule. The corpus contains 11331 sentences, about 350 thousand words. We extract sentences containing polysemes from the corpus to build our Chinese WSD dataset. We evaluate our model on the corpus and achieved good results.

The rest of this paper is structured as follows. In Section II, we introduce some related work about the WSD problem. Then in Section III, the strategy and the model are explained in detail. Afterwards, the experiments and results are presented in Section IV. We conclude the research in Section V.

2 Related Work

Recently, many works have been done on word sense disambiguation problem. One approach [3] presents an unsupervised vector-space model using global context and local context to represent different meaning of a word as different vectors. This work uses k-means clustering algorithm, and the results are not so ideal.

Another work [4] uses synonyms to replace original words in context to fake context. Then a naïve bayesian model is put forward to predict the meaning of a word. But the model isn't universal. Such a model can only classify one word's meanings. If a new word comes, new corpus should be gathered and a new model should be trained.

Most recently, there is another work [5] based on Bi-directional LSTM (Bi-LSTM). This model has the same disadvantage as the work above, i.e., lack of universality. Therefore, it is still open to design a framework for the universal WSD task.

3 Method

3.1 Problem Statement of WSD

Up to now, many strategies have been proposed on this problem. Some of them use context semantic information [3], and others take advantage of the additional information, such as a semantic dictionary [5]. However, none of them considered synonym as a resource to solve the problem.

A single word may have different synonyms corresponding to its different meaning. As in the example shown before, the word "zu" has many synonyms, among which "si wang" corresponds to "death" and "shi bing" corresponds to "solider". Therefore, if we replace the original word "zu" with its difference synonyms, the original sentence is converted into different sentences. Among these generated sentences, only one sentence is "legal", i.e., correct in semantic, because only one synonym is suitable in the sentence. For example, there is a Chinese sentence below:

Zhu Geliang sheng yu 181 nian, zu yu 234 nian.

If we replace the word "zu" with its different synonyms, we could get two generated sentences:

Zhu Geliang sheng yu 181 nian, si wang yu 234 nian.(1)

Zhu Geliang sheng yu 181 nian, shi bing yu 234 nian.(2)

One can observe that, sentence (1) is correct in semantics, because the word "si wang" is suitable in this sentence. Also, sentence (1) didn't change the meaning of the original sentence. In comparison, sentence (2) is not semantic "legal". Firstly, the word "shi bing" is not appropriate in this sentence. The sentence is not clear and coherent at all. Secondly, it misses the original meaning "death" in the original sentence. In a word, sentence (2) is a wrong sentence.

Assume that there is a corpus containing many sentences, each of which has a word to disambiguate. We call these sentences as "sentences to disambiguate". For any sentence to disambiguate, one can generate many sentences using the synonyms of the word being disambiguated. The count of the sentences depends on the number of meanings the word to be disambiguated has. Among the sentences generated, one is correct and the others are wrong.

As a result, we transform the WSD problem to a sentence classification task. That is, we need to justify whether the new sentence is legal or not after the interesting word in the sentence is replaced by other of similar meaning.

3.2 Sentence Classification Based on LSTM

Using the strategy we introduce above, we get a corpus with positive examples and negative examples, and the amount of them are the same. From another point of view, the positive and negative examples can be treated as two different kinds of data. If we can train a model having the ability to classify sentences into these two categories, the WSD problem is changed into a text classification problem.

In this paper, we propose a text classification model based on LSTM and MLP, which references the model in [5]. However, there are several problems in their model. Firstly, the model is not universal. Once the model is trained, it only has the ability to disambiguate one word. If another word come, it will be necessary to collect corpus related to the word and train the model anew. Moreover, the model is based on Bi-LSTM [6], which considers the characteristic of English, where there exists parenthesis and subordinate clause. So the preceding part or the following part of the text alone has complete semantic information. But considering the characteristic of Chinese and the strategy's difference to their work, we use LSTM to model a sentence to get the complete semantic information of it.

The model's structure is shown as figure 1. The first layer of the model is an embedding layer. In this layer, a word is modified into a vector which can be used to calculate in computer. In this paper, we use a state-of-the-art tool word2vec [7] to complete the work.

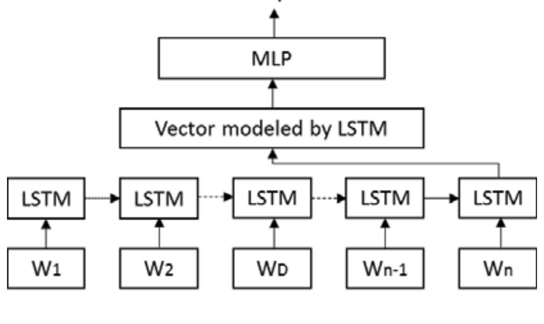

Figure 1. Model structure based on LSTM to classify texts.

The core idea of word2vec is that words similar in semantic have similar context. This tool has two training models. One is CBOW(Continuous Bag-of-Words) model and another is skip-gram model. The difference between these two models is that the training method are different. The skip-gram uses current word to predict the context, while the CBOW model uses the context to predict current word. It could be observed in its loss function. For the quality of the two models are close, so we just choose CBOW model as our training model. The corpus we used to train the model is Wikipedia Chinese corpus [8].

The second layer of our model is a standard LSTM layer. LSTM was used in various natural language processing tasks to model a text. It solve the gradient explosion and long-term dependence problem of general recurrent neural networks (RNN) [9] model. Consider the characteristic of Chinese and the need of complete semantic information of a sentence in our classification strategy, we didn't choose Bi-LSTM to model the preceding part and the following part of the text separately. As a result, we were only concerned about the output in last time step of a sentence.

The third layer is a MLP layer, which undertakes the task of classification. As we says above, the output of the second layer is a vector. And to perform a classification task, we need a sorter. The most common and effective method is a MLP sorter. We use one full connection hidden layer with 64 neurons and one output layer with one neuron in the MLP. The activation function is sigmoid and the output a classification result, which is the probability of the sentence to be legal. In another word, higher probability means the synonyms we replaced with the original word is more likely to be proper in the context.

For the training part, the loss function we use is binary cross-entropy [10] instead of the quadratic cost function. Because the traditional quadratic cost function has many disadvantages. For example, while using the back propagation algorithm to train the model, the larger the initial loss is, the slower the training proceeds. The binary cross-entropy loss function is as follows:

$$Loss = -\frac{1}{n}\sum_{x}[y\log a - (1 \quad y)\log(1 \quad a)]$$

In the function, α is the probability the model predicted, y is the real label of a sentence, x means all the sentences in the corpus.

Further, to overcome the problem of overfitting, we use the dropout technology [11] during the training process. The dropout means to ignore some weights randomly in the hidden layer while training. It prevents some situation in which weights are updated only under other specific characteristics and situations.

4 Experiment

4.1 New Dataset

There have been some datasets related to WSD task. The most popular datasets are the senseval [12] datasets. The senseval datasets includes Chinese WSD task dataset, in which the sentences are encoded by Hownet [13], a Chinese semantic dictionary. However, this dataset two disadvantages. Firstly, there are only hundreds sentences in the dataset. The amount is too small to process neural network training. Secondly, the encoding resource, Hownet, is based on morpheme instead of paraphrase or synonym. Considering this, we don't evaluate this model on this dataset.

Instead, we extract and build a new Chinese WSD dataset from HIT-CIR Tongyici Cilin (Extended) Corpus [2]. The dictionary has many rows of words. Words in the same row may have the same or close meaning. Especially, if there is only one word in a row, it's a monoseme. The dictionary uses different symbols to distinguish the difference. Table I shows the difference symbols of different senses. The symbol "=" in the end of a code means words in this row have exactly the same meaning, "@" means monoseme, and "#" means words in this row has similar meanings.

TABLE I. different symbols in HIT-CIR Tongyici Cilin (Extended)

Different symbols in codes	Example words
Ba01A02=	wuzhi zhi su
Ba01A03@	wan wu
Ba01B10#	daoti bandaoti chaodaoti

We only consider the words in rows with "=" as synonyms. As shown in the example above, words in rows with symbol "#" don't have the same meaning. The words "ban dao ti" and "chao dao ti" are not the same as "dao ti". So, we can't consider them as synonyms. We call the words row with "=" as "synonyms row".

If one word appears in different synonyms rows, we regard it as a polyseme. Further, in the corpus encoded in Cilin, if a sentence contains a polysemy we found by the rule above, it is necessary to disambiguate the polysemy word. In the end, we find 9799 sentences. For each sentence, we gather all the correct sentences generated based on the sentence in the corpus as the "is case set". In the same way, we gather all the wrong sentences as the "counter case set". To balance these two set, we only choose one wrong sentence corresponding to one original sentence in the corpus. As a result, these two sets constitute the word sense disambiguation corpus.

Finally, we build a Chinese word sense disambiguation corpus which contained 19598 sentences. Half of these sentences are correct and the other half are wrong.

4.2 Parameters Setting

The model we build have several parameters to set. The typical parameters are the size of word embedding, the rate of the dropout and the batch size during training. What's more, we also evaluate different RNN models, such as GRU(Gated Recurrent Network) [14] and simple RNN.

In the training step, we randomly pick 80% of the dataset as training set and the remaining 20% as the test set. In all the experiments below, the ratio remains the same. And for the record, all the results are obtained on the test set.

The experiments are all processed on a personal computer using Ubuntu 14.04 as operation system, Keras as code framework and having an 8GB internal memory, Intel i5 CPU and Nvidia Geforce 840m GPU. Moreover, we use the GPU to accelerate the training speed. The training time is measured on this situation.

4.3 Results and Discussion

Firstly we set the parameters as follows: embedding size is 256, batch size is 100, model is LSTM and dropout is 0.5. The training accuracy changes with the increase of the epoch. The result is as figure II.

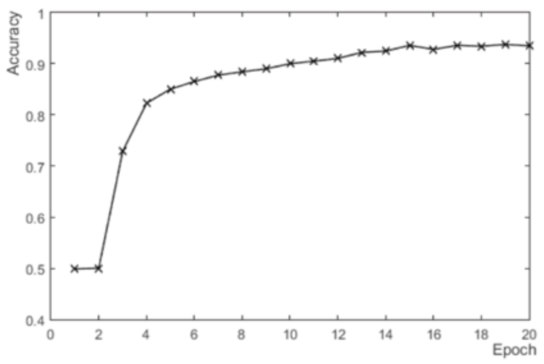

Figure 2. Training accuracy changes with the increse of epoch.

The result shows that the model has been trained well after 15 epochs. Afterward, the accuracy doesn't increase obviously. So in the experiment below, we set the epoch of the training as 15. It also shows that the strategy and the model we build is effective in the WSD task.

Secondly we evaluate different RNN models in order to prove the superiority of LSTM. The results are shown in Table II. Other parameters are also presented in the table.

Results on different RNN models

Model	Embedding size	Batch size	Dropout	Predicting accuracy
RNN	256	100	0.5	0.64
GRU	256	100	0.5	0.50
LSTM	256	100	0.5	**0.78**

As we can see, the LSTM model achieved the best result in the task, which proves the superiority successfully. In the experiments below, the LSTM model won't be changed.

Thirdly we evaluate the influence of the embedding size to the accuracy. We choose different embedding size as 64, 128 and 256. The results are shown in table III.

Results on different embedding size

Model	Embedding size	Batch size	Dropout	Predicting accuracy
LSTM	128	100	0.5	0.75
LSTM	256	100	0.5	**0.78**
LSTM	512	100	0.5	0.65

The result shows that the accuracy doesn't increase obviously with the growth of embedding size. It even decreases when the embedding size is large. The increase of calculation brings no profit. In the end, we choose 256 as our embedding size next.

Afterward, we change the batch size to find its influence on the accuracy. We use 50, 100, 150 as the value to evaluate. The result is concluded as table IV.

Results on different batch size

Model	Embedding size	Batch size	Drop out	Predicting accuracy	Training time
LSTM	256	50	0.5	0.49	1.5h
LSTM	256	100	0.5	**0.78**	10min
LSTM	256	150	0.5	**0.78**	8min

The result shows that it's important to use a batch size that is large enough, or the training time will be too long to endure. And if the batch size is too small, the predicting accuracy will also be bad. As a result we choose 100 as our batching size.

At last, we change the dropout rate to find a proper one that could complete the task best. We choose 0, 0.3, 0.5, 0.8 as candidates. Table V shows the results.

Results on different dropout rate

Model	Embed ding size	Batch size	Dropout	Predicting accuracy
LSTM	256	100	0	0.74
LSTM	256	100	0.3	0.75
LSTM	256	100	0.5	**0.78**
LSTM	256	100	0.8	0.73

One can see that if we set the dropout rate properly, it helps to increase the predicting accuracy. If we don't use this technology at all, the result may not become the optimal one.

In summary, the proposed model achieves better performance than the new state-of-the-arts. Also, an optimal parameter setting is provided after extensive experiments.

Conclusion

In this paper, we discuss a new method to solve the word sense disambiguation problem after transforming the original problem into sentence classification. An LSTM-based framework is proposed to perform the sentence classification. We also build a Chinese WSD dataset based on HIT-CIR Tongyici Cilin (Extended). Promising results is achieved in the dataset for the proposed method.

In future, we plan to 1) integrate the model with a multiple-class classifiers, so the meaning of a word can be output directly and 2) test the model for the English WSD task.

References

[1] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[2] Che, Wanxiang, Zhenghua Li, and Ting Liu. "Ltp: A chinese language technology platform." Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations. Association for Computational Linguistics, 2010.

[3] Huang, Eric H., et al. "Improving word representations via global context and multiple word prototypes." Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1. Association for Computational Linguistics, 2012.

[4] Yang, Zhizhuo, and Heyan Huang. "Chinese Word Sense Disambiguation based on Context Expansion." COLING (Posters). 2012.

[5] Kågebäck, Mikael, and Hans Salomonsson. "Word Sense Disambiguation using a Bidirectional LSTM." arXiv preprint arXiv:1606.03568 (2016).

[6] Schuster, Mike, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." IEEE Transactions on Signal Processing 45.11 (1997): 2673-2681.

[7] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

[8] Denoyer, Ludovic, and Patrick Gallinari. "The wikipedia xml corpus." International Workshop of the Initiative for the Evaluation of XML Retrieval. Springer Berlin Heidelberg, 2006.

[9] Williams, Ronald J., and David Zipser. "A learning algorithm for continually running fully recurrent neural networks." Neural computation 1.2 (1989): 270-280.

[10] Shore, John, and Rodney Johnson. "Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy." IEEE Transactions on information theory 26.1 (1980): 26-37.

[11] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[12] Mihalcea, Rada, Timothy Anatolievich Chklovski, and Adam Kilgarriff. "The Senseval-3 English lexical sample task." Association for Computational Linguistics, 2004.

[13] Dong, Zhendong, and Qiang Dong. "HowNet-a hybrid language and knowledge resource." Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003 International Conference on. IEEE, 2003.

[14] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).