HIT-IR-WSD: A WSD System for English Lexical Sample Task

Yuhang Guo, Wanxiang Che, Yuxuan Hu, Wei Zhang and Ting Liu
Information Retrieval Lab
Harbin Institute of technology
Harbin, China, 150001
{yhguo,wxche}@ir.hit.edu.cn

Abstract

HIT-IR-WSD is a word sense disambiguation (WSD) system developed for English lexical sample task (Task 11) of Semeval 2007 by Information Retrieval Lab, Harbin Institute of Technology. The system is based on a supervised method using an SVM classifier. Multi-resources including words in the surrounding context, the part-of-speech of neighboring words, collocations and syntactic relations are used. The final micro-avg raw score achieves 81.9% on the test set, the best one among participating runs.

1 Introduction

Lexical sample task is a kind of WSD evaluation task providing training and test data in which a small pre-selected set of target words is chosen and the target words are marked up. In the training data the target words' senses are given, but in the test data are not and need to be predicted by task participants.

HIT-IR-WSD regards the lexical sample task as a classification problem, and devotes to extract effective features from the instances. We didn't use any additional training data besides the official ones the task organizers provided. Section 2 gives the architecture of this system. As the task provides correct word sense for each instance, a supervised learning approach is used. In this system, we choose Support Vector Machine (SVM) as classifier. SVM is introduced in section 3. Knowledge sources are presented in section 4. The last section discusses the experimental results and present the main conclusion of the work performed.

2 The Architecture of the System

HIT-IR-WSD system consists of 2 parts: feature extraction and classification. Figure 1 portrays the architecture of the system.

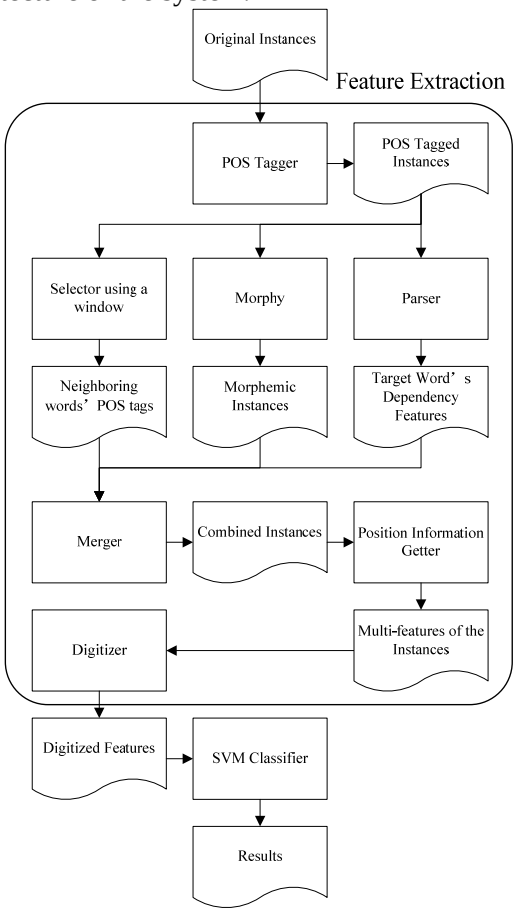

Figure 1: The architecture of HIT-IR-WSD

Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), pages 165–168,
Prague, June 2007.

Features are extracted from original instances and are made into digitized features to feed the SVM classifier. The classifier gets the features of training data to make a model of the target word. Then it uses the model to predict the sense of target word in the test data.

3 Learning Algorithm

SVM is an effective learning algorithm to WSD (Lee and Ng, 2002). The SVM tries to find a hyperplane with the largest margin separating the training samples into two classes. The instances in the same side of the hyperplane have the same class label. A test instance's feature decides the position where the sample is in the feature space and which side of the hyperplane it is. In this way, it leads to get a prediction. SVM could be extended to tackle multi-classes problems by using one-against-one or one-against-rest strategy.

In the WSD problem, input of SVM is the feature vector of the instance. Features that appear in all the training samples are arranged as a vector space. Every instance is mapped to a feature vector. If the feature of a certain dimension exists in a sample, assign this dimension 1 to this sample, else assign it 0. For example, assume the feature vector space is *<x1, x2, x3, x4, x5, x6, x7>*; the instance is *"x2 x6 x5 x7"*. The feature vector of this sample should be *<0, 1, 0, 0, 1, 1, 1>*.

The implementation of SVM here is libsvm[1] (Chang and Lin, 2001) for multi-classes.

4 Knowledge Sources

We used 4 kinds of features of the target word and its context as shown in Table 1.

Part of the original text of an example is "… *This is the <head>age</head> of new media , the era of …*".

Name	Extraction Tools	Example
Surrounding words	WordNet (morph)[2]	…, this, be, age, new, medium, ,, era, …
Part-of-speech	SVMTool[3]	DT_0, VBZ_0, DT_0, NN_t, IN_1, JJ_1, NNS_1
Collocation		this_0, be_0, the_0, age_t, of_1, new_1, medium_1, ,_1, the_1
Syntactic relation	MaltParser[4]	SYN_HEAD_is SYN_HEADPOS_VBZ SYN_RELATION_PRD SYN_HEADRIGHT

Table 1: Features the system extracted

The next 4 subsections elaborate these features.

4.1 Words in the Surrounding Context

We take the neighboring words in the context of the target word as a kind of features ignoring their exact position information, which is called bag-of-words approach.

Mostly, a certain sense of a word is tend to appear in a certain kind of context, so the context words could contain some helpful information to disambiguate the sense of the target word.

Because there would be too many context words to be added into the feature vector space, data sparseness problem is inevitable. We need to reduce the sparseness as possible as we can. A simple way is to use the words' morphological root forms. In addition, we filter the tokens which contain no alphabet character (including punctuation symbols) and stop words. The stop words are tested separately, and only the effective ones would be added into the stop words list. All remaining words in the instance are gathered, converted to lower case and replaced by their morphological root forms. The implementation for getting the morphological root forms is WordNet (morph).

4.2 Part-of-Speechs of Neighboring Words

As mentioned above, the data sparseness is a serious problem in WSD. Besides changing tokens to their morphological root forms, part-of-speech is a good choice too. The size of POS tag set is much smaller than the size of surrounding words set. And the neighboring words' part-of-speeches also contain useful information for WSD. In this part, we use a POS tagger (Giménez and Márquez, 2004) to assign POS tags to those tokens.

We get the left and right 3 words' POS tags together with their position information in the target words' sentence.

For example, the word *age* is to be disambiguated in the sentence of "… *This is the*

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvm/
[2] http://wordnet.princeton.edu/man/morph.3WN.html
[3] http://www.lsi.upc.es/~nlp/SVMTool/
[4] http://w3.msi.vxu.se/~nivre/research/MaltParser.html

<head>age</head> of new media , the era of …". The features then will be added to the feature vector are *"DT_0, VBZ_0, DT_0, NN_t, IN_1, JJ_1, NNS_1"*, in which *_0/_1* stands for the word with current POS tag is in the left/right side of the target word. The POS tag set in use here is Penn Treebank Tagset[5].

4.3 Collocations

Different from bag-of-words, collocation feature contains the position information of the target words' neighboring words. To make this feature in the same form with the bag-of-words, we appended a symbol to each of the neighboring words' morphological root forms to mark whether this word is in the left or in the right of the target word. Like POS feature, collocation was extracted in the sentence where the target word belongs to. The window size of this feature is 5 to the left and 5 to the right of the target word, which is attained by empirical value. In this part, punctuation symbol and stop words are not removed.

Take the same instance last subsection has mentioned as example. The features we extracted are *"this_0, be_0, the_0, age_t, of_1, new_1, medium_1"*. Like POS, *_0/_1* stands for the word is in the left/right side of the target word. Then the features were added to the feature vector space.

4.4 Syntactic Relations

Many effective context words are not in a short distance to the target word, but we shouldn't enlarge the window size too much in case of including too many noises. A solution to this problem is to use the syntactic relations of the target word and its parent head word.

We use Nivre et al., (2006)'s dependency parser. In this part, we get 4 features from every instance: head word of the target word, the head word's POS, the head word's dependency relation with the target word and the relative position of the head word to the target word.

Still take the same instance which has been mentioned in the las subsection as example. The features we extracted are *"SYN_HEAD_is, SYN_HEADPOS_VBZ, SYN_RELATION_PRD, SYN_HEADRIGHT"*, in which *SYN_HEAD_is* stands for *is* is the head word of *age*; *SYN_HEADPOS_VBZ* stands for the POS of the head word *is* is VBZ; *SYN_RELATION_PRD* stands for the relationship between the head word *is* and target word *age* is *PRD*; and *SYN_HEADRIGHT* stands for the target word *age* is in the right side of the head word *is*.

[5] http://www.lsi.upc.es/~nlp/SVMTool/PennTreebank.html

5 Data Set and Results

This English lexical sample task: Semeval 2007 task 11[6] provides two tracks of the data set for participants. The first one is from LDC and the second from web.

We took part in this evaluation in the second track. The corpus is from web. In this track the task organizers provide a training data and test data set for 20 nouns and 20 adjectives.

In order to develop our system, we divided the training data into 2 parts: training and development sets. The size of the training set is about 2 times of the development set. The development set contains 1,781 instances.

4 kinds of features were merged into 15 combinations. Here we use a vector (V) to express which features are used. The four dimensions stand for syntactic relations, POS, surrounding words and collocations, respectively. For example, *1010* means that the syntactic relations feature and the surrounding words feature are used.

V	Precision	V	Precision
0001	78.6%	1001	78.2%
0010	80.3%	1010	81.9%
0011	82.0%	1011	82.8%
0100	70.4%	1100	73.3%
0101	79.0%	1101	79.1%
0110	82.1%	1110	82.5%
0111	**82.9%**	1111	**82.9%**
1000	72.6%		

Table 2: Results of Combinations of Features

From Table 2, we can conclude that the surrounding words feature is the most useful kind of features. It obtains much better performance than other kinds of features individually. In other words, without it, the performance drops a lot. Among these features, syntactic relations feature is the most unstable one (the improvement with it is unstable), partly because the performance of the dependency parser is not good enough. As the ones with the vector 0111 and 1111 get the best perfor-

[6] http://nlp.cs.swarthmore.edu/semeval/tasks/task11/description.shtml

mance, we chose all of these kinds of features for our final system.

A trade-off parameter C in SVM is tuned, and the result is shown in Figure 2. We have also tried 4 types of kernels of the SVM classifier (parameters are set by default). The experimental results show that the linear kernel is the most effective as Table 3 shows.

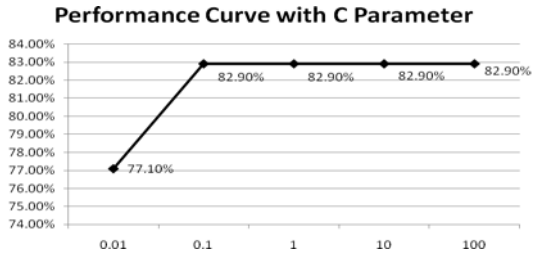

Figure 2: Accuracy with different C parameters

Kernel Function Type	Linear	Polynomial	RBF	Sigmoid
Accuracy	82.9%	68.3%	68.3%	68.3%

Table 3: Accuracy with different kernel function types

Another experiment (as shown in Figure 3) also validate that the linear kernel is the most suitable one. We tried using polynomial function. Unlike the parameters set by default above (g=1/k, d=3), here we set its Gama parameter as 1 (g=1) but other parameters excepting degree parameter are still set by default. The performance gets better when the degree parameter is tuned towards 1. That means the closer the kernel function to linear function the better the system performs.

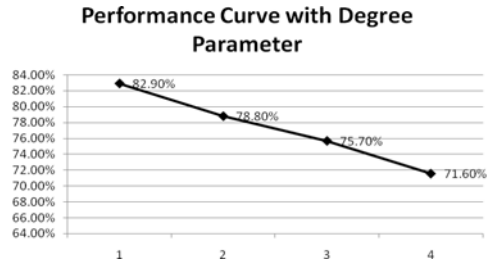

Figure 3: Accuracy with different degree in polynomial function

In order to get the relation between the system performance and the size of training data, we made several groups of training-test data set from the training data the organizers provided. Each of them has the same test data but different size of training data which are 2, 3, 4 and 5 times of the test data respectively. Figure 4 shows the performance curve with the training data size. Indicated in Figure 4, the accuracy increases as the size of training data enlarge, from which we can infer that we could raise the performance by using more training data potentially.

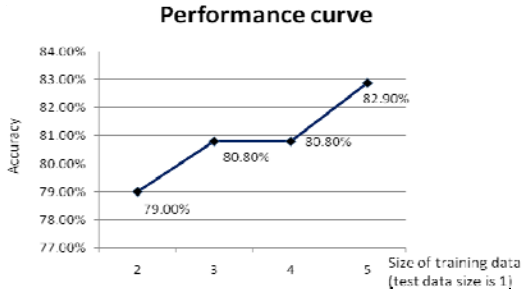

Figure 4: Accuracy's trend with the training data size

Feature extraction is the most time-consuming part of the system, especially POS tagging and parsing which take 2 hours approximately on the training and test data. The classification part (using libsvm) takes no more than 5 minutes on the training and test data. We did our experiment on a PC with 2.0GHz CPU and 960 MB system memory.

Our official result of HIT-IR-WSD is: micro-avg raw score 81.9% on the test set, the top one among the participating runs.

Acknowledgement

We gratefully acknowledge the support for this study provided by the National Natural Science Foundation of China (NSFC) via grant 60435020, 60575042, 60575042 and 60675034.

References

Lee, Y. K., and Ng, H. T. 2002. *An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation*. In *Proceedings of EMNLP02*, 41–48.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*.

Jesús Giménez and Lluís Márquez. 2004. *SVMTool: A general POS tagger generator based on Support Vector Machines*. Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04). Lisbon, Portugal.

Nivre, J., Hall, J., Nilsson, J., Eryigit, G. and Marinov, S. 2006. *Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines*. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*.