Word Sense Prediction Using Decision Trees

Heng Low Wee * and Min-Yen Kan †
Department of Computer Science,
School of Computing,
National University of Singapore

Abstract

Much studies have been done on Word Sense Disambiguation (WSD), but few touched on keeping the system light-weight and speedy. In this report, we describe how we create a light-weight and speedy system that is directed at keeping model size and time taken minimal. A light-weight WSD system can be implemented for real-time applications on web browsers and mobile devices. We adopted Decision Trees in our system, and demonstrated that we are able to reduce our model size to 1MB, and perform WSD at an average speed of 0.10 seconds per sentence.

1. INTRODUCTION

WSD is a key task in Natural Language Processing. It has many applications in computational linguistics, such as machine translation, text mining and information retrieval. It can also be applied within search engines such that the search results will be more relevant, when the results have the same intended meaning as the search query.

There are many WSD systems that can provide great accuracies in disambiguating words. However, most of them are also not publicly available, either for its applications or further research. Also, they are not suitable to be deployed as real-time softwares, such as for use on mobile devices, as the size of their training models are relatively large.

The motivation in this project, other than to encourage the usage of WSD applications, is to build WSD systems that are small, and responsive such that they are more suited to be deployed on web browsers and mobile devices.

We identified three goals for our system: Accuracy, Speed and Size of Model. We propose a system that is meant to perform WSD on small amount of text, like a sentence. One application for such a system is a language-assistance based tool to be integrated with web browsers, like a Firefox plugin, so that users can find out more information, like pronunciation & definition, about the words on the web page. WSD comes into the picture as we see the need for these information to be context-relevant. We must also keep the system responsive to maintain users' experience.

We want to keep the model size at minimal, while retaining a reasonable level of accuracy. We want it to be speedy and responsive. Also, we intend to reduce the amount of pre-processing prior to the prediction of word senses. Too many pre-processing features would require additional supporting files that incur additional space.

---

* Student
† Supervisor

2. RELATED WORK

In general, supervised WSD has always been attaining performance better than the other types of WSD systems, as they utilized large sense-tagged corpus as training model. However their WSD accuracies are only as good as how large and fulfilling their training models are. For the long term, we must consider unsupervised systems for their ability to perform independent of sense-tagged corpus. To encourage WSD applications in web browsers and mobile devices, we must consider portability, in other words its model size and time needed for the WSD process. For that, we have selected the following works to study in detail.

2.1. Word Sense Disambiguation Using Wikipedia

Wikipedia is a free, web-based and collaborative encyclopedia with most of its articles manually created and edited by the public. (Mihalcea, 2008) introduced an approach that exploits these manually tagged articles, building a sense tagged corpus from Wikipedia to be used for WSD.

The advantage of this approach is it exploits the dynamic nature of Wikipedia to create a source for corpus that will always be up-to-date and contains any new words or new senses. A disadvantage, however, is Data Inconsistency, when different users might refer to the same object by a different name. This approach has the flexibility to handle new entries, as they would likely appear on Wikipedia faster than any formal dictionary databases. But if the number of new words and senses grows as fast as the number of articles, there is a need to re-construct the sense-tagged corpus. Else, we may use the APIs on MediaWIki to send query requests to retrieve information about a target word and build a real-time WSD system with a dynamic lexical databases and corpora. Performance issues aside, this minimizes the need to reconstruct a corpus on an occasional basis.

2.2. Word Sense Disambiguation Using Dependency Knowledge

(Chen Ping, 2009) formulated the WSD problem into *weighted directed graphs*, and by simply computing values of the weighted nodes, it is effective in telling the dependencies between the words in a given text.

The authors' algorithm is effective and accurate in matching the dependency relations to determine the correct senses. However, the pre-processing and parsing processes take a lot of time. Then, the authors highlighted there is this concern regarding the dependency parser, Minipar causing some invalid dependency relations in the parse trees. Although the authors claimed that the WSD algorithm will minimize those erroneous relations, it was not explicitly defined how it actually did it.

2.3. Word Sense Disambiguation Using Noisy Channel Model

In this paper, (Yuret & Yatbaz, 2010) describes an unsupervised system that uses a generative probabilistic model for WSD. In their system, they used unlabeled data sets derived from publicly-available web pages instead of sense-tagged corpus. They demonstrated that the Noisy Channel Model can also be used for WSD. Their main contribution is the reduction of the WSD problem into the estimation of two distributions: the distribution of words that can be used in a given sense and in a given context.

In their experiments the authors compared their system to some of the best supervised and unsupervised systems. Their probabilistic approach outperforms all some semi-supervised systems that were compared with. In other words, their unsupervised WSD had

outperformed some of the semi-supervised ones, which should be considered a remarkable feat.

2.4. It Makes Sense

It Makes Sense (IMS) is a supervised English all-words WSD System introduced by (Zhong & Ng, 2010). The main contribution of IMS is being an open source system that allows users to customize it by integrating different preprocessing tools and additional features. Another contribution is that the system addresses the issue regarding the lack of sense-annotated training examples, which is critical in the overall performance of a supervised WSD system.

Regarding the performance, on a default configuration of modules, IMS attained state-of-the-art accuracies on all-words and lexical-sample tasks, with performance comparable to the top performing system compared to in the paper.

2.5. Using Decision Trees Of Bigrams To Predict Word Sense

(Pedersen, 2001) presents a corpus-based approach to WSD where a decision tree assigns a sense to an ambiguous word based on the *bigrams* that occur nearby. The author took this approach because surface lexical features like bigrams often contribute a great deal to disambiguation accuracy. In this paper, the author demonstrated that Decision Trees can be used as an accurate predictor for word senses.

2.6. Conclusion

From the above works, we have learnt that a supervised WSD system needs to have sufficient, if not abundant training examples in order to attain higher WSD accuracies. However, there is a constant need to update the training model used by the system. Furthermore, increasing the size of the training model only improves its performance by 3 to 4%(Yuret & Yatbaz, 2010). This is not cost effective as manual tagging is time consuming and costly.

In most papers related to WSD, little was mentioned regarding reducing the model size and time taken. We propose the idea of *portability*. By portability we refer to a system that is light-weight and speedy.

With the above in mind, we describe our approach as follow.

3. SYSTEM DESCRIPTION

Following a study on related works, we find the need to build a system, which is light-weight and speedy. For that, we populated the following characteristics that we want in our system:

1. Requires minimal pre-processing for training and test inputs
2. Training model must be small, and easy to interpret
3. Perform speedily

Decision Trees is one of the simplest predictive model and thus simple to interpret and understand. It requires little data preparations, and is able to have value even with little hard data and process large amount of data in a short time. With that, we decided to

implement our system using Decision Trees. We use the Weka[3] implementation of the J48 decision tree learner to generate the model files to help us predict word senses.

The intuition is that a sentence is the *environment* that helps determine the word senses of ambiguous words. We look at the surrounding words of a target word in the same sentence, such that if there exists a particular set of determining words, we predict that the target word would possess a specific word sense.

We used the SemCor1.7.1 corpus as our training data set and retrieved a total of **20450** *lemmas*. By *lemma* we refer to the root form of a word (E.g. root form of "ate" is "eat"). Predictions are based on the sense keys extracted. Since our approach predicts word senses by sentences, we parsed the training corpus and saved it into numerous sentences and then as an instance for that lemma if it contains that lemma.

For each lemma, we generated the model files with pruning to remove any irrelevant branches in the trees, reducing the size of the trees. There are **2261** trees that are *unary* and captures only one word sense. After some investigation, we found these lemmas had only one instance of occurrence in the corpus. For the other trees, their *depths* range from **5 to 49**, and the number of *leaf nodes* ranges from **2 to 25**.

Consider a sentence with only one ambiguous word. We use the rest of the sentence to make the prediction. Ignoring the order and repetition of words, an English sentence can be denoted as a set of English words, as such: $S_i = \{a_1, a_2, \ldots, a_k\}$. Therefore, for ambiguous a_j, $S - \{a_j\}$ is the *environment* that determines its word sense. We use it to generate a single data instance to test against our training data.

In the following sections, we describe our system and provide the test results iteratively. We used SemCor1.7.1 as the training corpus for the SensEval-2, SensEval-3 and SemEval all-words tasks. Our *Baseline* is the WordNet-Sense-One, which assigns the most frequently tagged sense as the answer. The results of the WSD accuracies are shown in Table 2.

3.1. Iteration 1 (DT v0.1)

In this iteration, a single-instance test file is generated for every testable word in the sentence. Then we run the prediction process to get the word sense. The only information we used is the occurrence of the neighboring words of a testable word, the *environment* as we defined.

We quickly noticed that the speed of the prediction process was very slow. On average, the time taken for each sentence can easily be **10 to 40 seconds**. Preliminary speed comparison with the IMS system showed that IMS performs at least 70% faster than our system. Also, we compared our model size with IMS's, and our system's model size, at **545MB**, was twice as large as IMS's, at **275MB**. We address this two issues in Iteration 2.

3.2. Iteration 2 (DT v0.2)

We omitted a lot of unimportant information and only used the actual *tree* itself for the prediction process. We extracted these trees and further compacted each tree into a single line. As a result our model is now a single text file at approximately **1 MB**. With this, we were able to keep our model size very small, and to fulfill one of our goals: Size of Model. Our prediction procedures were modified due to this change in model format. We dropped

[3]http://www.cs.waikato.ac.nz/ml/weka/

the usage of Weka. Instead, each node of a tree is a test condition. Prediction means testing against these nodes and eventually leading to a leaf node that provides a word sense. We evaluated this iteration of our system to ensure its consistency in accuracy, and also speed. In this new procedure, we processed *all* sentences from the three test corpus we tested against in approximately 2 minutes, averaging at about **0.10 seconds** per sentence. We reduced the time taken by at least **98%** and thus fulfuling the other goal: Speed.

	Before DT v0.2	DT v0.2
Model Size (MB)	545	**1**
Time Taken (secs per sentence)	10 to 40	**0.10**

Table 1. Comparison of model size and time taken

3.3. Iteration 3 (DT v0.3)

In the previous iteration, we did not lemmatize the attributes in our training data files. We speculated that if we lemmatize the attributes, we should get better results. Doing so also removes redundant attributes and nodes in the trees. For that we generated a new version of the model. In this new version, the size of the model was further reduced by **2%**, from 1,052,703 to 1,032,414 bytes. Zooming in to each tree, we have tree size reduction of up to **22%**.

We also added the *confidence* feature, whereby if the confidence in the prediction is too low, we will switch to using the first sense of WordNet. We evaluated this new iteration and we attained improvements for all the corpus we tested against (refer to Table 2). We demonstrated that even with a further reduction of model size, we were able to attain an improvement in WSD accuracy.

For the time taken to perform WSD, we evaluated our system against the IMS system. Clearly, with larger model files, IMS would take a longer time during the WSD process. To make comparison fair, the number of test sentences must be large enough to make the loading time less significant. We used all the sentences we parsed from the three test corpus to form a single test file of **789** sentences. Our system loads the model into memory every time it performs WSD on a new sentence, whereas only once for IMS. The time taken for this speed test is summarized into Table 3. Clearly, when performing WSD on large amount of text, IMS performs about **38%** faster than DT v0.3.

	SensEval-2 (%)	SensEval-3 (%)	SemEval Coarse-Grained (%)
Baseline (WNs1)	55.9	48.8	64.3
DT v0.1	48.5	45.5	57.8
DT v0.2	48.5	45.5	57.8
DT v0.3	49.2	46.2	58.3
IMS	68.2	67.6	82.6

Table 2. WSD accuracies on SensEval and SemEval all-words tasks

	IMS	DT v0.3
789 sentences	50.322 secs	81.148 secs
Average	0.0637 secs	0.103 secs

Table 3. Speed Test Experiment 1

	IMS	DT v0.3
62 sentences	17.636 secs	**7.611 secs**
Average	0.284 secs	**0.123 secs**

Table 4. Speed Test Experiment 2

However this is not the intended nature for our system. Our system is meant to only perform WSD on a single sentence. In order to illustrate how our system actually performs, we need to perform a second speed test. In this second speed test, we picked **62** random sentences from the test corpus. We ran these sentences through both system for **10** rounds each, and computed the average score for each round. For this speed test, we used the original DT v0.3 to compare against IMS. Making reference to the figures in bold in Table 4, our system is faster in this speed test, when the setting is only to test for small amount of text. The speed performance of our system fulfills one of our goals: Speed.

4. CONCLUSION

Even though there is much study made on WSD, few focused on introducing *portability* into the system. For that we proposed a light-weight and speedy system. We wish to implement such a system so that we can perform real-time WSD on platforms like web browsers, or even mobile devices.

We acknowledge that our system produces low WSD accuracies. However there is still plenty of room of improvement. There are many other surface lexical features that can applied into our system. Considering the potential for growth, our idea of a light-weight and speedy WSD system is definitely feasible.

SensEval-2 (%)	SensEval-3 (%)	SemEval-2007 Coarse-Grained (%)
67.6	65.6	84.7

Table 5. Corpus overlap with SemCor1.7.1 (with respective test corpus)

Additional post-experiment analysis shows that the amount of overlap of the training and test corpus affected the WSD accuracies. Comparing Table 2 and 5, we can see that the percentages in overlap are consistent with the performances in various test corpus. However it also showed the weakness of our system's inability to process *unknown* words to the system. An immediate course of action from this point might be to increase the amount of training data. The advantage in this move is that it would expose the system to more words and senses. However, increasing the size of training corpus would directly increase the size of the model and conflict with our goals. Therefore, in the longer term, we wish to introduce some *unsupervised* characteristics to the system, so that it is less dependent on back-end resources.

Our experiments showed that our system met our goals of being light-weight and speedy. This satisfies our intention to implement such a system into real-time applications like web browsers and mobile devices. We make reference to previous work done on DiCE Translator[4] prior to this system. As a Firefox plugin, DiCE's purpose is to provide bilingual translations for text on web pages to encourage language learning across a second language. Despite after extracting dictionary content from Wiktionary[5], DiCE lacks the ability to translate accurately according to the context. Integrating our system into DiCE, we plan for DiCE to be able to translate for relevantly, while still maintaining the overall responsiveness in using the tool.

We conclude that we had successfully created a system that is light-weight and speedy.

[4]https://addons.mozilla.org/en-US/firefox/addon/dice-translator/

[5]http://www.wiktionary.org/

5. REFERENCES

[1] Chen Ping, Ding Wei, C. B. . D. B. (2009). A fully unsupervised word sense disambiguation method using dependency knowledge. *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics on - NAACL '09*, (June), 2009, 28.

[2] Mihalcea, R. (2008). Using wikipedia for automatic word sense disambiguation, 2008.

[3] Pedersen, T. (2001). A decision tree of bigrams is an accurate predictor of word sense. *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, NAACL '01 (pp. 1–8), Stroudsburg, PA, USA, 2001: Association for Computational Linguistics.

[4] Yuret, D., & Yatbaz, M. A. (2010). The Noisy Channel Model for Unsupervised Word Sense Disambiguation. *Computational Linguistics*, *36*(1), March, 2010, 111–127.

[5] Zhong, Z., & Ng, H. T. (2010). It makes sense: A wide-coverage word sense disambiguation system for free text. *Proceedings of the ACL 2010 System Demonstrations*, ACL '10 (pp. 78–83), Morristown, NJ, USA, 2010: Association for Computational Linguistics.