WORD SENSE DISAMBIGUATION USING WORDNET

by

Anıl Can Kara & Oğuzhan Yetimoğlu

Submitted to the Department of Computer
Engineering in partial fulfillment of
the requirements for the degree of
Bachelor of Science

Undergraduate Program in Computer Engineering
Boğaziçi University

Spring 2019

WORD SENSE DISAMBIGUATION USING WORDNET

APPROVED BY:

Prof. Tunga Güngör .....
(Project Supervisor)

DATE OF APPROVAL: 16.05.2019

ABSTRACT

WORD SENSE DISAMBIGUATION USING WORDNET

The concept of sense ambiguity means that a word which has more than one meaning is used in a context and it needs to be clarified that which sense is actually referred. Word sense disambiguation (WSD) is the concept of identifying which sense of a word is used in a sentence or context.

Sense disambiguation is a problem that can be overcomed easily by complex structures of human brain. In computer sciences, however, it can be solved by using appropriate algorithms according to the application of words. In computer sciences, this problem is one of the most important and current issues in Natural Language Processing (NLP).

Word sense disambiguation is a very important task in natural language processing applications such as machine translation and information retrieval. In this paper, different approaches to this problem are described and summarized, and another method for Turkish using WordNet is proposed. In this method, Lesk algorithm is used but it is extended in a way that it exploits the hierarchical structure of WordNet.

ÖZET

TÜRKÇE WORDNET İLE KELİME ANLAMININ BELİRLENMESİ

Anlam belirsizliği kavramı, doğal dillerde çokça görülen, bir kelimenin birden fazla anlama sahip olması durumudur. Sözcük anlam belirsizliği giderme ise bu birden fazla anlama sahip olan sözcüğün cümle içinde hangi anlama geldiğini belirleme kavramıdır.

Anlam belirsizliği, insanlarda karmaşık düşünme yapıları sayesinde bağlamdan yola çıkarak üstesinden kolayca gelinebilen bir sorunken bilgisayar bilimlerinde, ancak sözcüğün uygulanma şekline göre uygun algoritmaların kullanılması ile çözümlenebilir. Bilgisayar bilimlerinde ise bu sorun Doğal Dil İşleme (DDİ) alanlarında ele alınmakta olan önemli ve güncel konular arasında yer almaktadır.

Kelime anlamı belirsizliği, makine çevirisi ve bilgi alma gibi doğal dil işleme uygulamalarında çok önemli bir görevdir. Bu yazıda, bu soruna farklı yaklaşımlar anlatılmıştır ve bu yaklaşımlar özetlenmiştir. Aynı zamanda Türkçe için Wordnet kullanılarak anlam belirsizliğini gideren başka bir yöndem ileri sürülmüştür. Bu yöntemde Lesk algoritması kullanılır, ancak WordNet'in hiyerarşik yapısını kullanacak şekilde genişletilir.

TABLE OF CONTENTS

LIST OF ACRONYMS/ABBREVIATIONS

NLP	Natural Language Processing
WSD	Word Sense Disambiguation

1. INTRODUCTION AND MOTIVATION

Word sense disambiguation refers to the process of determining the correct sense (meaning) of a word in a given context. For example, one may want to know if the word *fare (mouse)* in Turkish is meant to be an animal or a computer device in a sentence. This is automatically done by human beings because natural languages are parts of our lives and it is very natural to us to extract the meanings of the words in different contexts. However, this is a very hard task for computers. First of all, for a computer, words are just a sequence of characters and the characters are just a sequence of bits which may vary according to the encoding scheme used. Therefore, there is a need for clever algorithms to tell the computer to understand the meanings of the words. This also requires sufficient data which is WordNet, an hierarchical lexical database, in our case.

Our project topic is a crucial part of natural language processing, because knowing the sense of a word in a given context is very helpful in lots of applications such as machine translation and information retrieval. Without any sense disambiguation, it is very difficult to translate a sentence which contains the word *fare* from Turkish to any other language. Similarly, search engines have to account for the correct sense of a word during the information retrieval, otherwise a person who looks for a mouse for her computer may find herself looking at the pictures of animals.

The motivation behind such project is the importance of word sense disambiguation and the lack of extensive works in this field in Turkish language. We believed that we could combine well-known approaches to the problem together and put our ideas on top of them to create an accurate system.

2. STATE OF THE ART

While we were doing research, we have seen that there are a lot of papers regarding word sense disambiguation task. In this chapter, we will summarize these approaches briefly and we will demonstrate the papers that we have examined.

There are different approaches to the problem and various methods to use. These are mainly knowledge-based, corpus-based and machine learning-based approaches. Knowledge-based methods rely on some predefined knowledge databases such as WordNet. Corpus-based approaches use available texts as examples and they use these examples in classification process. Machine learning methods are usually used with corpus-based approaches and they aim to decide the correct sense of the word by making use of supervised or unsupervised machine learning algorithms. Since we have used WordNet, our work can be considered as knowledge-based. However, all the methods described above can be combined and we can also make use of other methods in a future work.

During the preparation process, we have read various articles in this field to have a deeper understanding about the topic. Below, you can find the summaries:

2.1. Word Sense Disambiguation Using WordNet Relations

[4]It starts by defining Lesk Method which is based on counting the common words in the synsets of the target word and the other words that are in the context, both in bag of words representations. However, it turns out that applying only Lesk Method for word sense disambiguation results in relatively low accuracy. That's why they exploit WordNet hierarchy to get a higher accuracy. For this purpose, they use WordNet hyponym/hypernym relationships in a way that the number of common words is also being calculated for "parent" synsets, not just for the synsets that are given by the words in the given sentence, which they call "base synsets". "Child" synsets are not included because some experiments show that they do not contribute to the accuracy.

They assign a weight for each synset which is basically inversely proportional to its synset level. In other words, the words in a synset get a weight inversely proportional to the synsets' distance from base synset. For example, base synset gets a weight of 1 and the parent synset gets a weight of 0.5 and so on. Furthermore, they also include every word's definition in the word bags, not just the word's themselves to improve the accuracy. Finally, they calculate a score for each sense of the target word by calculating the common words between the bags described above and multiplying them by the weights of words. The sense that has the highest score is selected as the proposed sense for the target word.

2.2. Unsupervised Word Sense Disambiguation Using WordNet Relatives

[9]This work is based on utilizing WordNet relations as much as possible. The key idea that they follow is as following: Given a context and a target word, first, they find the relatives of the target word's senses in the WordNet hierarchy. Since the distant relatives are more irrelevant than the close ones, they consider each relative separately instead of putting them into one bag. Then, for each relative, they calculate the co-occurrence of that relative with the context words. For this purpose, they create a co-occurrence matrix of all pairs of words in a large corpus. Finally, they conclude that the word that has the highest co-occurrence can be substituted with the original word and therefore, it is the proposed sense of the target word.

2.3. Determining Senses for Word Sense Disambiguation in Turkish

[7]In this work, they start by annotating the correct senses of the word "git" manually in seven world classics. Then, they argue that using pseudowords may help a lot during the disambiguation process. Their idea is to concatenate the words that have no direct relationships between them such as "haber" and "ağaç" to form the pseudoword "haberağaç" and then whenever there is a sentence that contains either of these words, it will be considered as it contains the word "haberağaç". In their opinion, this approach helps to obtain a sense tagged corpus for later stages. However, they don't explain how it helps and what they do in the next stages. That's why this

article was actually not that helpful to improve our understanding in this area.

2.4. Word Sense Disambiguation for Turkish Lexical Sample

[8]This research focuses on four Turkish words in detail, instead of creating a working system for all words. These four words are "bas","gül", "kır" and "yüz" each of which has lots of different senses in Turkish. Article starts by summarizing general methods that is being used for word sense disambiguation and continues by describing their work. Firstly, they collect sentences from seven books written by Tolstoy, Turgut Özakman, Barbara Taylor and Marlo Morgan. Then, they collect the senses for the four words given above from TDK and eliminate some senses by merging the ones that have very similar meanings. From the seven books selected, they pick 100 sample sentences for each sense of the word "bas" (300 in total since there are 3 senses at the end) and apply the similar approach for the other three words. After getting all these sentences, they label the senses of the words manually. Collecting the number of senses for the four words selected is struggling in this work, because in different dictionaries, the number of senses are different for these words. They deal with this problem by eliminating the senses which are similar in meaning. Using a parser called Zemberek which is designed for Turkish, they parse the sentences and decide the features that affect word sense disambiguation. Although it says supervised machine learning algorithms and Naive Bayes is used, there are no details about how they are implemented which is the most important lacking of this paper.

2.5. A New Semantic Similarity Measure Evaluated in Word Sense Disambiguation

[1]This research actually aims for finding a similarity measure between two words. Since they test the similarity measure that they propose in a word sense disambiguation task, the approach that they follow to measure similarity may help us in our project. The article starts with some other similarity measures. It describes a very important algorithm which is called Maximum Relatedness Disambiguation, or alternatively, Adapted Lesk Algorithm. In this algorithm, a window of size n which contains the tar-

get word in the middle is selected and all the senses of the target word are compared with the words in the window. Depending on the similarity measure followed during this process, scores for each sense are calculated and the sense that has the highest score is proposed as the correct sense of the target word. Researchers use this algorithm to evaluate their similarity measure. The idea behind their measure is the combination of the length and specificity. By length, they mean the shortest path between the words in the WordNet taxonomy, and by specificity, they mean how specific the given word is. A word is specific if it is in the deeper nodes in WordNet and a word is abstract if it is in the upper nodes of the hierarchy. Combining these ideas, they measure their precision and recall and they get accurate values.

2.6. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone

[5]In 1986, Michael Lesk proposed the algorithm that would later be called the Lesk algorithm in the article. The reason that led Lesk to find this approach is that previous procedures use existing dictionaries and can only use immediate context to process any text. Lesk algorithm is basically, retrieves from dictionary all sense definitions of the words to be disambiguated, determines the definition overlap for all possible sense combinations, and chooses senses that lead to highest overlap. Although this algorithm does not produce best results at the beginning, in the hybrid algorithms where the lesk algorithm is used, much more successful results can be achieved.

2.7. An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet

[2]The paper written by Satanjeev Barenjee and Ted Pedersen presents that rather than using standard dictionary for Lesk algorithm, a large lexical English database WordNet can be selected. They adapted Lesk algorithm to the WordNet corpus. This adapted Lesk algorithm takes a word as an input, and outputs the WordNet meanings of the word's occurrences in the text.

First step of the method starts with taking window of contexts (2n+1) from left and right of the word of interest. Afterwards, all possible senses of the tokens are taken from WordNet corpus. With this data, all combination scores are evaluated. The final output is the best score.

Since our goal is similar, this technique would be a guide to reach our goal.

2.8. A WordNet-based Algorithm for Word Sense Disambiguation

[6]This paper presents the WordNet-based algorithm to disambiguate word senses. This algorithm aims at word sense disambiguation of noun objects in a text.

The approach targets 2 main parts. One of them is usage of the critical relationships between words in WordNet. The other is "word sense disambiguation heuristic rules" stand on semantic relationships which is mentioned above.

2.9. Survey of Word Sense Disambiguation Approaches

[10] In this survey, Xiaohua Zhou and Hyoil Han summerizes view about word sense disambiguation approaches. The paper gives a general view of a knowledge about sources (corpuses etc.), techniques (approaches like unsupervised and supervised approach and their types) used for word sense disambiguation. Also it mentions about these approaches' complexities, pros and cons, performance etc.

This paper was particularly useful in terms of providing an overview of word sense disambiguation and its approaches.

2.10. Building a Wordnet for Turkish

[3] This article summarizes the development process of the Turkish WordNet. History of the construction process of the Turkish WordNet is presented briefly. Applications based on the Turkish WordNet were mentioned and introduced. Also, semantic

relations used in WordNet like synonyms, hyperonmys were introduced. The article has a practical tips and links section that can be helpful.

This article was useful for better understanding and perceiving of Turkish WordNet.

As it can be seen in the article summaries above, it is easy to find papers related to word sense disambiguation. They all describe different approaches which enlarges the understanding of the reader. However, some of the articles do not contain sufficient information about implementation details and they only give brief information about the topic. Still, they are very informative in the sense that they provide a broad perspective.

The most important shortcoming of the state-of-the-art is the lack of high quality researches in Turkish. Most of the papers related to word sense disambiguation are introducing the approaches for English and for other languages such as Korean.

3. METHODS

One of the most popular algorithms in word sense disambiguation is Lesk algorithm which was first introduced by Michael E. Lesk in 1986. The basic idea behind Lesk algorithm is to calculate scores for each candidate sense by counting the common words between the synsets of these candidate senses and the synsets of the words in the given context, and to pick the sense that has the highest score as the proposed one. Although this idea is not completely sufficient for word sense disambiguation, it provides a very good starting point because it is very simple, intuitive, easy to implement and flexible.

Since we have a lexical database like WordNet, it is possible to extend Lesk algorithm. For this purpose, we have exploited the hierarchical structure of WordNet to modify Lesk algorithm in a way that the words in hypernyms are integrated into the Lesk algorithm. In the upcoming sections, our methodology is described in detail step by step. We have used Python to implement all the functionalities described in this chapter.

3.1. Preprocessing

First step is to apply some preprocessing on the user input. The reason is, it is very likely that there will be lots of punctuations, uppercase letters and stopwords which do not contribute to the disambiguation process. Therefore, preprocessing is a necessity and its steps are listed below:

- Tokenization
- Removal of 284 stopwords
- Case folding
- Removal of punctuations
- Lemmatization using Zemberek

Zemberek is an open-source parser that was designed for Turkish. Since all the words in WordNet are in their base form, the words in the input strings are converted into their base forms using Zemberek to make them consistent with WordNet. It is important to note that Zemberek is not perfect and it sometimes generates incorrect results, but in general, it works good enough.

3.2. Extended Lesk Algorithm

After the preprocessing part, the algorithm that disambiguates the sense of the word specified as the target word is applied on the strings generated by preprocessor. In the algorithm, there are two important parameters which play an important role on the accuracy values. These parameters are the window size and the number of levels. Window size refers to the number of words contiguous to the target word that are going to be considered in the disambiguation algorithm. For example, if the window size is 1, it means that only the words which have distance of 1 to the target word in the given sentence are involved. The other parameter, number of levels, refers to the number of hypernym levels that are going to be taken into account. For instance, if the level is set to 3, the algorithm goes up 3 levels in the hypernym hierarchy of WordNet and it stops after the third iteration. We changed these parameters to get the best results which will be discussed in the next chapter. With the ideas of window size and number of levels, the algorithm proceeds as follows:

(i) The algorithm is based on a famous approach in NLP applications called bag of words representation. In this representation, the order of the words does not matter and a text is treated as a bag of words. It is a good fit for our project to count the common words.

(ii) We start by defining N different bags where N is the number of senses of the target word that we want to disambiguate. Into these bags, we put all the words that are in the synset of the corresponding sense and the words in that synset's definition. This constitutes the *0th level* of the algorithm because at this stage, hypernyms are not considered yet.

(iii) Similarly, we define another bag for nearby words. However, instead of creating

separate bags for each sense, we put all the senses together this time because we do not need to know the exact sense that the word belongs to. The only functionality of this bag is to make it possible to calculate scores for each sense which are clearly split up in the first bag.

(iv) With two bags in our hands, the next step is to enlarge them by adding hypernyms. The number of hypernyms to be taken into account is decided by the parameter called level, as described above. Note that the effect of hypernyms on the scoring scheme decreases as we climb up in the WordNet hierarchy and it is inversely proportional to the level number. For instance, detected common words in *level 0* adds score of 1 to the related sense, but if that common words are in, let's say, *level 1*, the score is 0.5 since we climbed one more level in the hierarchy. This makes sense because the concepts become more generic as the level increases, and the commonality of generic terms should not be a strong sign of similarity in that case. For example, the words *bez* and *araba* are both *objects*, but this does not mean that they are closely related.

(v) Finally, according to the scoring scheme clarified above, scores for each sense are calculated and the one that has the highest score is proposed as the correct sense of the word in the given context. A very simplified version of the algorithm is given on the next page as pseudo-code.

3.3. Data Collection and Labeling

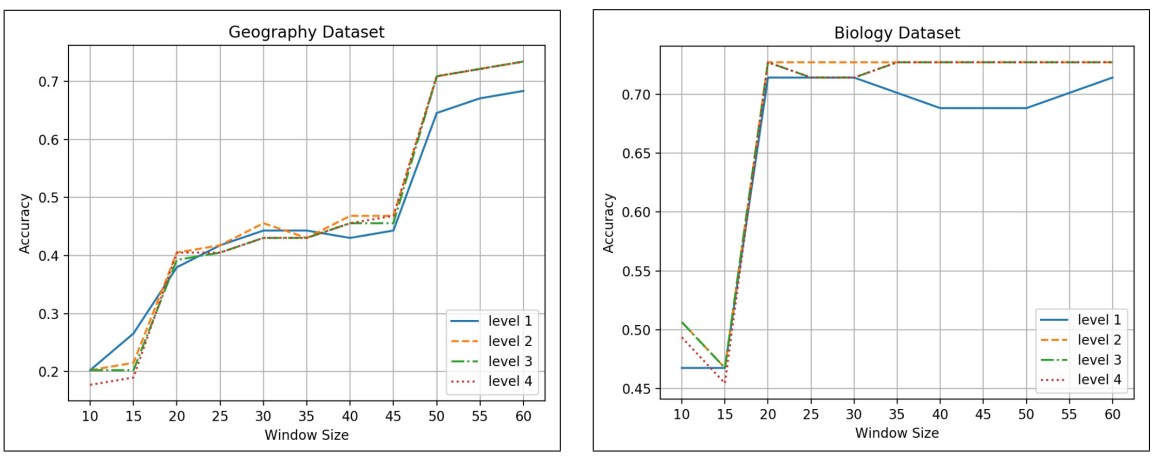
To test the algorithm, we collected two different texts from the books of Ministry of National Education (MEB). As our first dataset, we used a high school geography book. From this book, we extracted a text containing 413 words and we defined 52 words as target words by looking at their availability in WordNet. Then, we labeled correct senses of these target words one by one. Similarly, we also used the biology book of MEB for high school students and extracted 1042 words with 22 target words in it. The results of the evaluation is discussed in the next chapter.

Algorithm 1 Simplified Disambiguation Algorithm

```
1:  procedure DISAMBIGUATE
2:      target ← getTargetFromUser()
3:      context ← getContextFromUser()
4:      tokens ← preprocess(context)
5:      senses ← getSenses(target)
6:      synsetWords ← getSynsetWords(target)
7:      definitionWords ← getDefinitionWords(target)
8:      initialize bagForEachSense
9:      initialize bagForNearbyWords
10:     for each sense in senses do
11:         bagForEachSense[sense] ← synsetWords + definitionWords
12:     for each nearby word do
13:         bagForNearbyWords ← getSynsetWords(word) + getDefinitionWords(word)
14:     while level < maxLevel do
15:         for each sense of target word do
16:             bagForEachSense[sense ← getHypernymWords(sense)
17:         for each sense of nearby words do
18:             bagForNearbyWords ← getHypernymWords(sense)
19:         weight ← 1/level
20:         updateScores(bagForEachSense, bagForNearbyWords, weight)
```

4. RESULTS

As an indicator of success, we have calculated accuracies on two different texts in this project. As a first attempt, we used geography dataset as described in the previous chapter. While we were labeling the correct senses on this dataset, we chose as many words as we can. For example, even if some words' senses are not clearly distinguished and not well-defined in WordNet, we somehow included them by selecting one of the senses which makes sense to us. However, it is important to emphasize that this is a bit misleading and accuracies are less than expected in this case, because in reality, there is no reason to reject other senses just because their definitions are not well-defined in WordNet. But when the algorithm proposes one of them as a result, we treat it as a wrong prediction which decreases accuracy misleadingly. Therefore, in our second attempt in which we used biology dataset, we have chosen the target words in a way that no such problem exists. For this purpose, we have selected a word as a target word only if its senses are clearly distinguished in WordNet. We believe that this dataset gives more realistic accuracy values than the first one which are compared below. In both biology and geography datasets, we excluded the words that are either not in WordNet or have only one sense during the accuracy calculation. According to these specifications, the results are depicted with respect to the variable window size in the figure below. Colors represent the different level selections for the parameter that controls how many hyperyms are considered in the algorithm.



(a) (b)

Figure 4.1. Accuracies

The first observation from the plots is that accuracy improves with the increasing size of window up to a certain point, and then it stabilizes. For the biology dataset, window size of 20 looks enough while geography dataset needs a window size of 60 to be stabilized. Since we find the biology results more realistic for the reasons explained above, we can say that keeping the window size between 20 and 30 is optimal for the purposes of the sense disambiguation task.

Secondly, we observe that increasing the level does not contribute after the second level. The main reason is that the concepts in the WordNet hierarchy get too generic after some point as we climb up. Therefore, setting the number of levels to 2 and looking only 2 levels up in hypernym hierarchy is the optimal solution.

Overall, the best performance that we get is acquired by setting the window size to 20 and the number of levels to 2, and the accuracy in this case is 72.73%. In addition to these, we have also considered adding hyponyms and meronyms besides hypernyms into the bags of words, but they did not improve the performance and even made it worse in some cases. That makes sense because if we consider hyponyms, for instance, they are the words that are more specific than the original word, and counting the words that are too specific misguides the algorithm.

5. CONCLUSION AND DISCUSSION

To conclude, although there were some difficulties regarding the weaknesses of Turkish WordNet and Zemberek, we believe that we get pretty high accuracy values at the end. The algorithm performs quite well in most of the examples and when it fails to generate correct output, it is very likely that it is because there is not much information about the related words in WordNet. For instance, some of the synsets have English definitions instead of Turkish. Therefore, even if we include the definitions inside the bag of words, some common words cannot be detected.

As an example, the word *neyzen* is defined as *person who plays the musical instrument 'ney'* which is in English, but the word *müzik* is defined as *duygu ve düşünceleri tek sesli veya çok sesli olarak anlatma sanatı* which is in Turkish. As a result, even if there are some commonalities between these two concepts, our algorithm may fail to find related words such as *müzik* and *musical* and treat them as totally different concepts. Since there is no consistency between Turkish and English definitions in WordNet and they are used randomly, there is no way to avoid such problem. Although the algorithm still works well, we argue that it would generate even more accurate results with a more structured and systematic WordNet.

6. FUTURE WORK

As future work, the first thing that needs to be considered is the integration of another source of information to the project. WordNet is quite powerful because of its ability to represent the relationships between the words, but lots of the synsets do not have a definition and some of the senses of the words are not well-defined. Thus, an external source such as Turkish Language Society (TDK) glossary can be very helpful to increase the success of the algorithm.

In addition, since Turkish is an highly agglutinative language, suffixes can give us some important hints during the disambiguation process. Therefore, it is also worth considering to make use of suffixes to decide the correct sense of a word. However, this needs an extensive search and thinking on Turkish grammar and it has to be implemented carefully, otherwise it may spoil the flow of the original algorithm.

To conclude, both of the options mentioned above are remarkable and may increase the performance of the algorithm significantly. We would be more than happy if we have an opportunity to work on these suggestions one day.

□

REFERENCES

1. Altıntaş, E., E. Karslıgil and V. Coşkun, "A new semantic similarity measure evaluated in word sense disambiguation", *Proceedings of the 15th Nordic Conference of Computational Linguistics (NODALIDA 2005)*, pp. 8–11, University of Joensuu, Finland, 2006.

2. Banerjee, S. and T. Pedersen, "An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet", A. Gelbukh (Editor), *Computational Linguistics and Intelligent Text Processing*, pp. 136–145, 2002.

3. Bilgin, O., "Building a Wordnet for Turkish", , 2004.

4. Fragos, K., Y. Maistros and C. Skourlas, "Word Sense Disambiguation using WordNet Relations", *In Proc. of the 1st Balkan Conference in Informatics, Thessaloniki*, 2003.

5. Lesk, M., "Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone", *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pp. 24–26, 1986.

6. Li, X., S. Szpakowicz and S. Matwin, "A WordNet-based Algorithm for Word Sense Disambiguation", *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pp. 1368–1374, 1995.

7. Orhan, Z. and Z. Altan, "Determining Senses for Word Sense Disambiguation in Turkish", *International Enformatika Conference, IEC'05, August 26-28, 2005, Prague, Czech Republic, CDROM*, pp. 187–192, 2005.

8. Özdemir, V., "Word sense disambiguation for Turkish lexical sample", *Fatih University in partial fulfillment of the requirements for the degree of Master of Science*,

2009.

9. Seo, H.-C., H. Chung, H.-C. Rim, S. H. Myaeng and S.-H. Kim, "Unsupervised word sense disambiguation using WordNet relatives", *Computer Speech & Language*, Vol. 18, No. 3, pp. 253 – 273, 2004.

10. Zhou, X. and H. Han, "Survey of Word Sense Disambiguation Approaches", *FLAIRS Conference*, 2005.