WORD SENSE DISAMBIGUATION FOR TURKISH LEXICAL SAMPLE

by

Vildan ÖZDEMİR

July 2009

WORD SENSE DISAMBIGUATION FOR TURKISH LEXICAL SAMPLE

by

Vildan ÖZDEMİR

A thesis submitted to

The Graduate Institute of Sciences and Engineering

Of

Fatih University

in partial fulfillment of the requirements for the degree of

Master of Science

In

Computer Engineering

July 2009

Istanbul, Turkey

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Assist. Prof. Tugrul YANIK
Head of Department

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist. Prof. Zeynep ORHAN
Supervisor

Examining Committee Members

Assist. Prof. Zeynep ORHAN

Assist. Prof. Tuğrul YANIK

Assist. Prof. Nurgül ÖZCAN

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Sciences and Engineering.

Assoc Prof. Nurullah ARSLAN
Director

WORD SENSE DISAMBIGUATION FOR TURKISH LEXICAL SAMPLE

Vildan ÖZDEMIR

M. S. Thesis - Computer Engineering
July 2009

Supervisor: Assist. Prof. Zeynep ORHAN

ABSTRACT

Word Sense Disambiguation (WSD) is the process of disambiguation of the sense of a word when the word has more than one sense using the position of the word in a sentence, and the relation of the word with other words in the sentence.

In this thesis, the process of WSD is explained for the words in Turkish which has sense disambiguity by choosing appropriate features and algorithms. In the rich language structure of Turkish, four example words which have more than one sense, have been selected and the WSD study has been performed for these words. Due to the lack of sense annotated text to be able to do these types of studies, first the data was collected composed of sentences containing the sample words chosen. Then the features that discern the sense of the word have been identified, Supervised Learning algorithms have been applied to the data, and the results obtained using evaluation methods have been interpreted.

For sense disambiguation, NaiveBayes, KStar, SimpleCart and Bagging algorithms have been used in the test processes performed. Furthermore the features for the words have been identified that are believed to be effective in the study. The effect of the selected algorithms and features has been shown. The points that will be useful to be considered for the studies to come in this area have been mentioned.

Keywords: Natural Language Processing, Word Sense Disambiguation, Supervised Learning Algorithms.

TÜRKÇE SÖZCÜK ÖRNEKLERİ İÇİN SÖZCÜK ANLAMLARININ BELİRGİNLEŞTİRİLMESİ

Vildan ÖZDEMIR

Yüksek Lisans Tezi – Bilgisayar Mühendisliği
Temmuz 2009

Tez Yöneticisi Yrd. Doç.Dr. Zeynep ORHAN

ÖZ

Sözcük anlamını belirginleştirme, kelimenin birden fazla anlama sahip olması durumunda, yer aldığı cümledeki konumu ve diğer kelimelerle ilişkisine göre anlamının belirginleştirilmesi işlemidir.

Bu çalışmada Türkçe metinler içerisinde anlam belirsizliğine sahip olan sözcükler için uygun özellik ve algoritmaların seçilerek, anlam belirginleştirme işleminin nasıl yapıldığı anlatılmaktadır. Türkçenin zengin dil yapısı içerisinde birden fazla anlama sahip dört örnek kelime seçilerek bu kelimeler için anlam belirginleştirme çalışması yapılmıştır. Bu çalışmaların yapılacağı işaretlenmiş metinlerin eksikliği sebebiyle öncelikle seçilen örnek kelimeleri içeren cümlelerden oluşan veriler toplanmıştır. Daha sonra çalışmada seçilen kelimeler için anlamın ayırt edilmesini sağlayan özellikler belirlenmiş ve Öğreticili Öğrenme algoritmaları verilere uygulanmış, değerlendirme yöntemleri ile elde edilen sonuçlar değerlendirilmiştir.

Yapılan test işlemlerinde anlam belirginleştirmesi için NaiveBayes, Kstar, SimpleCart ve Bagging algoritmaları kullanılmıştır. Ayrıca kelimeler için çalışmada etkili olabilecek özellikler belirlenmiştir. Seçilen bu algoritmaların ve özelliklerin ne kadar etkili oldukları ortaya konmuştur. Bundan sonraki çalışmalar için göz önünde bulundurulması faydalı olabilecek noktalara genel olarak değinilmiştir.

Anahtar Kelimeler: Doğal Dil İşleme, Kelime Anlamının Belirginleştirilmesi, Öğreticili Öğrenme Algoritmaları.

DEDICATION

To my family

ACKNOWLEDGEMENT

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I am grateful to my adviser, Assist. Prof. Dr. Zeynep ORHAN, for giving me the opportunity to study and conduct research for constitution of this thesis. Her knowledge and her logical way of thinking have been of great value for me. I appreciate her patience, understanding and encouragement throughout my thesis study.

I would like to thank my friend Zeliha GÖRMEZ for the support she has given during my thesis.

I would like to give my special thanks to my husband Özgür and my whole family whose patient love and endless help enabled me to complete this work.

TABLE OF CONTENTS

ABSTRACT ... III

ÖZ ... IV

DEDICATION ... V

ACKNOWLEDGEMENT ... VI

TABLE OF CONTENTS ... VII

LIST OF TABLES ... X

LIST OF FIGURES ... XII

LIST OF SYSMBOLS AND ABBREVIATIONS ... XIII

CHAPTER 1 ... 1

INTRODUCTION ... 1

1.1 GENERAL PURPOSE ... 1
1.2 OVERVIEW OF WORD SENSE DISAMBIGUATION ... 3
1.2.1 Word Sense Disambiguation and Application Areas ... 3
1.2.2 Necessary Types of Information in Word Sense Disambiguation ... 4
1.2.3 Problems Encountered in Word Sense Disambiguation ... 4
1.3 GENERAL VIEW OF WSD METHODS ... 5
1.3.1 Machine Learning Approaches ... 5
1.3.1.1 Supervised Learning Approaches ... 6
1.3.1.2 Unsupervised Learning Approaches ... 6
1.3.2 Important Studies in Word Sense Disambiguation ... 7

CHAPTER 2 11

IMPLEMENTATION 11

2.1 DESIGN OF THE CORPUS 11
2.2 SELECTION OF THE FEATURES EFFECTING THE WORD SENSE 18
2.3 PROJECT ZEMBEREK 22
2.4 ZEMBEREK PARSER PROGRAM 26
2.5 WEKA PROJECT 34
2.5.1 The WEKA Explorer 34
2.5.2 Methods Used In WEKA Project 38
2.5.2.1 Statistical Methods 38
2.5.2.2 Decision Trees: 39
2.5.2.3 Instance Based Classification 40
2.5.2.4 Bootstrap Aggregating Methods 41
2.5.3 Evaluation Methods 42

CHAPTER 3 45

TEST AND RESULTS 45

3.1 ACCURACY 45
3.1.1. Comparison Between Words 46
3.1.2. Comparison Regarding Performance Assumption Techniques 47
3.2 CONFUSION MATRIX AND PRECISION-RECALL RESULTS 52
3.3 TESTS REGARDING FEATURE SELECTION 59

CHAPTER 4 63

DISCUSSION AND CONCLUSION 63

4.1 PROBLEMS ENCOUNTERED 63
4.2 RESULTS AND FUTURE STUDIES 64

REFERENCES 67

APPENDIX A 71

LIST OF TABLES

TABLE

2.1 The Senses of the Word “Basmak” in TDK. 12
2.2 The Senses of the Word “Gül” in TDK. 13
2.3 The Senses of the Word “Kır” in TDK. 14
2.4 The Senses of the Word “Yüz” in TDK. 15
2.5 The Senses of the Words and Their Markings 16
2.6 Sample Sentences 17
2.7 Total Number of Sentences, Word Senses, POS 17
2.8 The Word Types and Suffixes 20
2.9 The Word Types and Suffixes for a Sample Sentence. 22
2.10 Zemberek Output for a Sample Sentence 25
2.11 Main function of Zemberek Parser Pseudocode 29
2.12 Sub-function of Zemberek Parser Pseudocode 30
2.13 Output as Arff Files 31
2.14 A Sample of Arff Files 33
2.15 List of Options in NaiveBayes 39
2.16 List of Options in SimpleCART 40
2.17 List of Options in KStar 41
2.18 List of Options in Bagging 42
2.19 Table of Confusion Matrix 43
3.1 Number of Sentences per Words and Their Baselines 45
3.2 The Accuracy Results of Algorithms (%) 46
3.3 Confusion Matrix of the Word “Bas” 52
3.4 The Confusion Matrix of the Word “Bas” for NaiveBayes – CV 53
3.5 Confusion Matrix of the Word “Gül” 54

3.6 Confusion Matrix of the Word “Kır” 55
3.7 Confusion Matrix of the Word “Yüz” 56
3.8 Precision Recall Values for the Word “Bas” 57
3.9 Precision Recall Values for the Word “Gül”. 58
3.10 Precision Recall Values for the Word “Kır” 58
3.11 Precision Recall Values for the Word “Yüz” 59
3.12 Selected Features 60
3.13 Type of Representation 61
3.14 Precision Results 62
3.15 Recall Results 62

LIST OF FIGURES

FIGURE

1.1 Example of Visual Uncertainty.......2
2.1 Zemberek Screen Shot.......24
2.2 Zemberek Parser Screen Shot.......26
2.3 Zemberek Parser Input Screen for the Word “Gül”.......27
2.4 Zemberek Parser Input screen for the Word “İçin”.......27
2.5 The Zemberek Parser Input screen for the Other Option.......28
2.6 Interface of WEKA.......35
2.7 WEKA Clasification Tab.......36
2.8 The List of Algorithms in WEKA.......37
3.1 The Accuracy Results of Word "Bas".......49
3.2 The Accuracy Results of Word “Gül”.......49
3.3 The Accuracy Results of Word “Kır”.......50
3.4 The Accuracy Results of Word “Yüz”.......50
3.5 Cross Validation Accuracy Results.......51
3.6 Train-Test Accuracy Results.......51

LIST OF SYSMBOLS AND ABBREVIATIONS

SYMBOL/ABBREVIATION

AC	Accuracy
AI	Artificial Intelligence
CM	Confusion Matrix
CV	Cross-Validation
IE	Information Extraction
IR	Information Retrieval
ML	Machine Learning
MT	Machine Translation
NB	NaiveBayes
NLP	Natural Language Processing
P	Precision
POS	Part of Speech
R	Recall
TDK	Turkish Language Association (Türk Dil Kurumu)
TP	True Positive Rate
TT	Train-Test
WSD	Word Sense Disambiguation

CHAPTER 1

INTRODUCTION

1.1 GENERAL PURPOSE

Artificial Intelligence (AI) can be defined as complex mental activities, such as reasoning, inference and generalization, which are known as human specific features, to be fulfilled by computers or machines under computer control. The first studies in this area were: Alan Mathison Turing that allowed the breaking of German passwords during the Second World War, the finite state machines and the algorithm called the Turing machine. Another study carried out by Turing was the Turing test[1]. The test consisted of a person asking questions to a person and a smart machine via keyboard. If the questioner cannot distinguish the person and the machine from each other in a reasonable period of time, Turing was concluding that the machine had a kind of intelligence. A program called ELIZA, designed by Joseph Weizenbaum, followed these studies. In this program, the purpose was to provide interaction between the computer and the person by establishing certain conversations (Weizenbaum, 1966). The purpose of this program was to make simple analysis with the machine and establish conversation between the patient and the psychoanalyst. The studies in the area of Artificial Intelligence formed a lot of subfields. Natural Language Processing (NLP) is an area researched as a subfield of Artificial Intelligence and Linguistics.

The basic purpose of Natural language processing is to provide communication between the computers and a natural language. Natural language processing brings together the theories, methods and technologies developed in various fields such as artificial

[1] Turing Test: http://www.turing.org.uk/turing/scrapbook/test.html

intelligence, theory of morphological languages, theoretical linguistics, for this reason, the subjects for research increased and it became a discipline studied alone.

The major problem encountered during natural language processing is ambiguity. Ambiguity is the property of being ambiguous, where a word, term, notation, sign, symbol, phrase, sentence, or any other form used for communication, is called ambiguous if it can be interpreted in more than one way[2]. The concept of ambiguity is not only a problem for natural languages processing but also it is a problem encountered in many areas. For example in the Figure 1.1 someone can see a woman or a saxophone player (Schrater and Sundareswara, 2006). Both of them can be perceived from the pictures and this is not a problem.

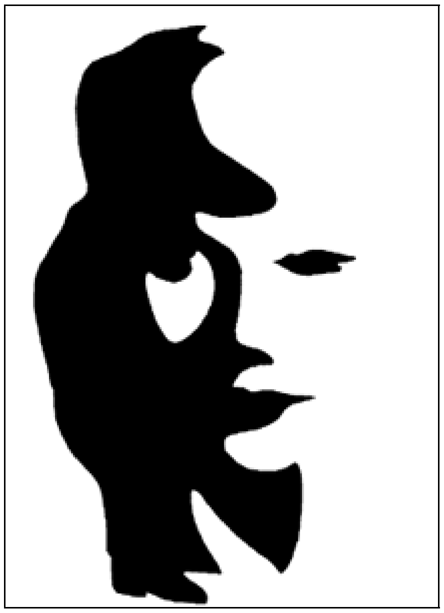


Figure 1.1 Example of Visual Uncertainty

We are often confronted with the words, which have more than one meaning in our daily life. If the word has more than one meaning or it can be understood in two or more possible ways then we can say it is ambiguous.

A sentence is structurally ambiguous if the grammar assigns it more than one possible parse (Jurafsky and Martin 2000). For example;

2 Ambiguity: http://en.wikipedia.org/wiki/Ambiguity

I know whom Ayla knows.

This sentence is ambiguous since it can be interpreted in two ways. Either I am acquainted with the same people as Ayla is, or I know who Ayla's acquaintances are.

The lexical ambiguity of a word or phrase consists in its having more than one meaning in the language to which the word belongs. For example:

Note means “A musical tone" or "A short written record."

Or another one;

Lie means "Statement that you know it is not true" or "present tense of lay: To be or put yourself in a flat position."

Generally people do not ask which meaning of the word is used in most cases. Most of the time their consciousness will easily find in the context of the meaning of expression by using where the use of meaning. The aim of the Word Sense Disambiguation (WSD) study is eliminating these ambiguities with a computer.

1.2 OVERVIEW OF WORD SENSE DISAMBIGUATION

1.2.1 Word Sense Disambiguation and Application Areas

One of the most important areas of study in WSD is Machine Translation (MT). If a word has more than one meaning, doing the correct translation is one of the application areas of WSD. WSD is required for lexical choice in MT for words that have different translations for different senses (Agirre and Edmonds, 2006).

Information Retrieval (IR) is another application area of WSD studies. Ambiguity has to be resolved in some queries. For instance, when the information related to the *"deniz"* is scanned through the documents, some sentences that involve the verb *"yüzmek”* can be found in these documents. However, the words that mean "*yüz"* as the number or human face should not be found

Information Extraction (IE) is another area of application of WSD. In many applications, text analysis should be made accurately. For instance, if the word "fare" is used in a text related with the computers, the sense to be perceived should be the tool that facilitates computer use, not an animal.

WSD and lexicography can work in a loop, with WSD providing rough empirical sense groupings and statistically significant contextual indicators of sense to lexicographers, who provide better sense inventories and sense- annotated corpora to WSD (Agirre and Edmonds, 2006).

Studies such as correct articulation of speech with stress and tones, lower/upper case correction in the text, etc are carried out within the scope of WSD (Orhan, 2006).

1.2.2 Necessary Types of Information in Word Sense Disambiguation

There have been some studies carried out on the types of information to be used and their effects on WSD for selecting the right sense of the target word in WSD studies. In line with these studies, it was observed that the following types of information should be used in WSD studies (Ide and Veronis, 1998; Agirre et al., 2001; Hirst, 1987).

- Determining part of speech of the words in a sentence
- Understanding the relations between the root and the word derived from the root
- Semantic word collocations formed by frequent use of two words together
- Determining the use of area of the word in the event that the text, in which the word is used, in a special context
- Using the word with multiple senses in different sentences with associative words
- Not only identifying the word with its own sense but also as sense classes for example determining the selective features such as identifying the sense with the help of the subject in the sentence
- The frequency of the usage of the senses of the words

Generally, these types of information are necessary for WSD.

1.2.3 Problems Encountered in Word Sense Disambiguation

Some problems are encountered in WSD studies. In general, these may be listed as follows:

- The first thing to consider in determining the sense of a word is the location of the word in a sentence. The locations of the word in the sentence, its grammar structure or the word with some other words are important features that effect determination of the word sense. However, it is not known which of these methods are most effective since the studies are carried out in a limited area (Orhan and Altan, 2005).
- Large scales of data are obtained by searching all senses of a word and preparing training data. It is a long and difficult process to carry out all necessary tests using these data. Moreover, when the words are examined, the number of senses found on one dictionary may not be found in another. In this case, the word selection of the dictionary as source or the online sources are very important and affect the study results (Basili et al., 1997).
- The word having lots of meaning or few meanings, or selecting words which are easy to distinguish and difficult to distinguish in applications, not being able to provide certain criteria for evaluating the results at certain standards in each study.

1.3 GENERAL VIEW OF WSD METHODS

1.3.1 Machine Learning Approaches

A major focus of Machine Learning (ML) research is to automatically learn to recognize complex patterns and make intelligent decisions based on data. It provides information about the past information and observations being in the future. In ML algorithms used for WSD studies, the main approach is to train a certain part of the words to be sense disambiguated, and find the correct group of sense of the words which have more than one sense. The data to be trained is called "input". According to Jurafsky and Martin, the input data is processed as follows:

- The original context may be replaced with larger or smaller segments surrounding the target word.
- Often some amount of stemming or more sophisticated morphological processing is performed on all the words in the context

- Less often, some form of partial parsing or dependency parsing, is performed to as certain thematic or grammatical roles and relations (Jurafsky and Martin, 2000).

As a result of these processes, a set of features that belong to the language related with the learning activity is obtained. For instance, the features of the words located left and right to the target word, the features of the root and suffixes of the target word.

ML algorithms are grouped into 2 as supervised learning and unsupervised learning algorithms.

1.3.1.1 Supervised Learning Approaches

Supervised learning is a ML technique for learning a function from training data. The training data consist of pairs of input objects and desired outputs. In supervised learning algorithms, there are inputs used in WSD whose instances are selected in advance and categorized according to their features. The systems output is, determining the accurate group of incoming data in accordance with the selected features. Examples of supervised learning algorithms may be Bayseian Classifiers (Duda and Hart, 1973), decision trees (Quinlan, 1986), Nural Networks (Rumelhart et al., 1986), logic learning systems (Mooney, 1995).

1.3.1.2 Unsupervised Learning Approaches

An unsupervised learning algorithm does not treat any particular attribute of its input instances as the target to be learned (Blau and McGovern, 2003).

In unsupervised learning methods, there are no pre-labeled instances. Clusters are established using the known word senses. Known words are placed in the clusters. The words, whose senses are unknown, are placed with a metric called the "similarity metric". The best-known algorithm of this method is Agglomerative Clustering (Jurafsky and Martin 2000).

In this algorithm, at the beginning every term forms a cluster of its own. Then the algorithm iterates over the step that merges the two most similar clusters still available, until one arrives at a universal cluster that contains all the terms.

There are different approaches for defining the distance between clusters that distinguishes the different algorithms.

According to Witten and Frank the approaches are as follows; Single linkage agglomerative clustering takes the intergroup dissimilarity to be that of the closest pair. Complete linkage agglomerative clustering takes the intergroup dissimilarity to be that of the furthest pair. Group average clustering uses the average dissimilarity between the groups. Centroid metric calculates the distance between cluster centroids. Medoid metric calculates the distance between cluster medoids (Witten and Frank, 2000).

1.3.2 Important Studies in Word Sense Disambiguation

One of the most important studies in WSD studies is the Senseval project[3]. Many computer programs have been developed regarding the WSD studies. The purpose of Senseval project is to evaluate the strengths and weaknesses of the program for different words and different types of languages. The first study carried out within this scope was Senseval 1.

Senseval is the first public and multi-participant WSD evaluation study. The target was to carry out the Senseval 1 study on English, French and Italian, however, the participants worked on English mostly due to the funds.

The dictionary and the corpus used within the scope of Senseval project was established based on the HECTOR study, which was developed in early 1990s. HECTOR is the database where a dictionary is related to a corpus. The reason of preferring this database was the cost, because, it both decreases the time and cost required for manual labelling and it has the features necessary for Senseval (Kilgarriff and Rosenweig, 2002).

In the studies carried out prior to Senseval, selection of the instances was up to the person or the program. However, in Senseval, layered random instances were selected, rather than a simple random selection.

Two types of data were used in Senseval (Kilgarriff and Rosenweig, 2002). These are the dry run training data and evaluation-test data. In the evaluation, the markers determined sentence senses of the words in the corpus first, and then the participant guessed the same

[3] Senseval: http://www.senseval.org/

words' senses. It was considered as "correct" if the senses were completely the same, "partially correct" if the senses matched partially, and "wrong" if the senses didn't match.

Three types of studies were carried out for 12 different languages in Senseval 2 workshop. These were carried out for the following:

- All words for the languages Czech, Dutch, English, Estonian (The studies have been made for all the words in these languages)
- Lexical sample/LS for the languages Basque, Chinese, Danish, English, Italian, Japanese, Korean, Spanish, Swedish (The studies have been made some sample words in these languages)
- Translation for the Japanese language.

A similar evaluation method as the Senseval 1 was used. The evaluation process used in Senseval 1 was used here as well, with some small changes. An important result obtained from Senseval 2 was the comprehension of the importance of obtaining an accurate sense classification and list.

Senseval 3 study was further developed based on the experience and background obtained from the previous studies. Senseval-3 included 14 different tasks for core word sense disambiguation, as well as identification of semantic roles, multilingual annotations, logic forms, and sub categorization acquisition.

- All words for the languages English, Italian
- Lexical sample/LS for the languages Basque, Chinese, English, Italian, Catalan, Romanian, Spanish, Swedish
- Automatic subcategorization acquisition
- Multilingual lexical sample
- WSD of WordNet glosses
- Semantic Roles
- Logic Forms
- Semantic roles for Swedish

SemEval 2007 is the 4th and the latest International Workshop on Semantic Evaluations. The committee selected 18 tasks to be part of SemEval 2007. These were carried out for the following:

- Cross Language Information Retrieval
- Evaluating Word Sense Induction and Discrimination Systems
- Classification of Semantic Relations between Nominals
- Multilingual Chinese-English Lexical Sample Task
- Word-Sense Disambiguation of Prepositions
- Coarse-grained English all-words
- Metonymy Resolution at Semeval-2007
- Multilevel Semantic Annotation of Catalan and Spanish
- English Lexical Substitution Task for SemEval-2007
- English Lexical Sample Task via English-Chinese Parallel Text
- Turkish Lexical Sample Task
- Web People Search, Affective Text
- TempEval: A proposal for Evaluating Time-Event Temporal Relation Identification,
- Evaluation of wide coverage knowledge resources
- English Lexical Sample
- English SRL and English All-Words Tasks
- Arabic Semantic Labeling
- Frame Semantic Structure Extraction [4].

When the studies carried out for Turkish in NLP field are considered morphological analyzer study is one of the first studies for Turkish in this field (Köksal, 1976). The other studies in this field are carried out by Jorge Hankamer (Hankamer, 1986) on Turkish morphology analysis in USA, Albert Stoop's study on morphology between Dutch and Turkish in the Netherlands (Stoop, 1987), Kemal Oflazer’s analyzer that used the two level morphology approach for Turkish morphological analysis (Oflazer, 1994), ELIZA programs Turkish adaptation (Aytekin et al., 1994), semantic and lexical researches (Kardes, 2002; Bozsahin, 2002; Demir, 2003), information inference and document classification processes (Pembe, 2004; Özgür, 2004).

Within the scope of WSD studies carried out for Turkish, a Turkish conceptual dictionary was prepared as a part of the BalkaNet[5] project at Sabancı University. Turkish

[4] SemEval: http://nlp.cs.swarthmore.edu/semeval/

[5] BalkaNet: http://www.hlst.sabanciuniv.edu/TL

WordNet[6] Project is a part of BalkaNet Project, which aims at the design and development of a multilingual lexical database for the Balkan languages Turkish, Greek, Bulgarian, Czech, Romanian, and Serbian with individual monolingual wordnets.

Up to this point, an introduction to NLP, a general view of WSD application areas, methods, and important works about WSD is given. The remainder of this thesis is organized as follows:

In the next chapter, it will be discussed the target words selected to be used in the study, their senses, and the features necessary for WSD and the features that are selected. In introduction general information regarding the WSD algorithms is given, detailed information will be provided in the next chapter regarding the particular WSD algorithms that will be used. Information will be given about the programs used and developed in the study, and finally, the necessary evaluation methods for evaluating the results in accordance with the selected algorithms will be explained.

In chapter 3, the details about our tests and results are demonstrated. Some discussion topics and the conclusion can be found in chapter 4.

[6] WordNet: http://wordnet.princeton.edu/

CHAPTER 2

IMPLEMENTATION

In this section, the methods and approaches followed during the process of disambiguation process of word sense for Turkish words will be mentioned.

The approach for WSD of this study consists of the following steps:

- Scanning appropriate data sources for the sample sentences used.
  - The designation of Turkish sample homophone words which shall be used in this work and determining senses of these words.
  - Collecting sentences including sample words, which were designated from the data sources scanned.
- In the sentences collected, defining the features of the word sense of the target word for in a certain sentence to be able to use it in learning methods.
- The examination of the program to be used in the research and writing the computer program where necessary.
- Choosing the appropriate program and algorithm for classification of the data collected and examination of the methods used by evaluating the results.

2.1 DESIGN OF THE CORPUS

The research has started with collecting sentences including words, of which the WSD shall be determined. The process of collecting such data took place in several ways. The sentences were acquired either from the world classics and other Turkish books uploaded to the Internet in e-book format, or the target word was searched in the Internet and sentences including this word were examined. In this context, the selected works were as follows:

- *Tolstoy: Efendi İle Uşağı (Master and Man), Diriliş (Resurrection), Anna Karenina*
- *Barbara Taylor: Yasak İlişki (A Secret Affair),*
- *Marlo Morgan: Bir Çift Yürek (Mutant Message Down Under),*
- *Turgut Özakman – Şu Çılgın Türkler.*

In determining the words to be used in WSD, the common senses of the words were chosen with care. The connotations were ignored and in phrases with more than one word sense, the most common word senses were chosen. Again, in our selection, not only one Part of Speech (POS), but many were selected, thus the relation between the word sense and types were examined.

In this research, the word roots, *yüz, bas, gül, kır* were chosen as the sample words. Since the senses of the bases of these words in this research were examined as verbs and nouns, the senses of these words were examined in TDK (Turkish Language Association). For example, in the word "bas", only its sense as a verb was traced. The senses of the word basmak in TDK were listed in Table 2.1.

Table 2.1 The Senses of the Word "Basmak" in TDK.

Sense Number	**Sense Definition**
1	Vücudun ağırlığını verecek bir biçimde ayak tabanını bir yere veya bir şeyin üzerine koymak
2	Bir şeyi, üzerine kuvvet vererek itmek
3	Örtmek, bürümek, kaplamak
4	Bir kimse bir yaşa girmek
5	Duman, sis vb. çevreyi kaplamak, çökmek
6	Uygunsuz vaziyette yakalamak
7	Çoğaltılması gereken bir yazı yapıtını basım yoluyla çoğaltmak
8	Basım işini gerçekleştirmek
9	Kümes hayvanları kuluçkaya yatmak.

If one examines the sense of the verb “basmak”, he/she notices that some of those senses are less commonly used and some of them build phrases with more or less similar senses. Hence, the senses number 4 and 9 in the table were removed and the senses number 5 and 6 were combined as sense groups similar to each other and marked as the first word sense group in the research. The first and the second senses had also similar meanings, thus they were also combined as a whole and marked as the second sense of the word “basmak”. The 7th and the 8th senses were also similar, thus they were marked as the 3rd sense of the word “basmak”. So, the word “basmak” was traced in sentences and if the word found was based upon one of those three senses, the sentences including these words were collected as collection texts.

In this research, the word senses and signs are listed in Table 2.5**.** Based upon the word senses listed in this table for word “bas”, 100 sample sentences for all three senses were chosen and 300 sentences were prepared. In four of the sentences collected for the second word sense the word bas is to be seen twice. Therefore, the number of bas words including the second word sense is 104 in total.

When one makes a research for the word “gül” in TDK, he/she shall notice that two main groups as nouns and verbs are useful for WSD. The word senses of the words “gül//gülmek” are listed in Table 2.2.

Table 2.2 The Senses of the Word “Gül” in TDK.

Sense Number	**Sense Definition**
1	Gülgillerin örnek bitkisi *(Rosa).*
2	Bu bitkinin katmerli, genellikle kokulu olan çiçeği.
3	Kısrakların tüylerinde beliren doğurma işareti.
4	Kişi Adı
5	İnsan, hoşuna veya tuhafına giden olaylar, durumlar karşısında, genellikle sesli bir biçimde duygusunu açığa vurmak:
6	Mutlu, sevinçli zaman geçirmek, eğlenmek, hoşça vakit geçirmek
7	Dikkati çekecek derecede hoş ve sıcak görünmek

When the word senses in Table 2.2 are examined, the 5^{th}, 6^{th}, and 7^{th} word senses are similar to each other, thus they were combined and marked as the first word sense. Similarly, also the word senses of the 1^{st} and 2^{nd} words were combined and marked as the second word sense of the word "gül". This situation can be seen in Table 2.5**.** Other word senses were ignored. In the research, 200 sample data were chosen for the 1^{st} and 2^{nd} word senses.

The other word that was researched is the word kır. After necessary examinations, it is grouped in two main groups; noun and verb. The word senses of the words "kır/kırmak" are listed in Table 2.3.

Table 2.3 The Senses of the Word "Kır" in TDK.

Sense Number	**Sense Definition**
1	Beyazla az miktarda siyah karışmasından oluşan renk
2	Şehir ve kasabaların dışında kalan, çoğu boş ve geniş yer:
3	kır, basık dağ, açık yer
4	Mayalanmış hamur
5	Sert şeyleri vurarak veya ezerek parçalamak
6	İri parçalara ayırmak.
7	Dileğini kabul etmeyerek veya beklenmeyen bir davranış karşısında bırakarak gücendirmek, incitmek
8	Sıvışmak, uzaklaşmak

Based upon the senses found in the dictionary, two main word sense groups can be seen for the base "kır". Thus when we searched for the target word "kır", the senses and markings listed in Table 2.5 were done. For the 1^{st} and 2^{nd} meaning of this word, 100 sample sentences in total were collected and a database of 200 sentences was acquired.

The last word that was chosen for this research is the word "yüz". The word "yüz" is used in 3 different forms in Turkish language. It is split into 3 main groups as noun, number, and verb. Thus the word senses of the word "yüzmek" was examined in the TDK dictionary and the results were listed in Table 2.4.

The more commonly used word senses one, two and five are the ones will be used for our research. The word senses and the markings in the table are shown in Table 2.5. For each word sense of the word “yüz”, 100 sentences were collected and a database of 300 sentences in total was acquired. Then again, in some sentences the word “yüz” was used twice thus, the number of target words for the first sense was 103, for the second sense it was 100 and for the third it was 101.

Table 2.4 The Senses of the Word “Yüz” in TDK.

Sense Number	**Sense Definition**
1	Doksan dokuzdan sonra gelen sayının adı.
2	Başta, alın, göz, burun, ağız, yanak ve çenenin bulunduğu ön bölüm, sima, çehre, surat
3	Birinin görüle gelen veya umulan hoşgörürlüğüne güvenilerek gösterilen cüret
4	Yan, taraf
5	Bir yapının dışa bakan düşey yüzeylerinin her biri
6	Kol, bacak, yüzgeç vb. organların özel hareketleriyle su yüzeyinde veya su içinde ilerlemek, durmak
7	Derisini çıkarmak, derisini soymak
8	Çok akmak

Some of the word senses in Table 2.5 have been grouped and then classified as a sense. In NLP, this is called “coarse grained”. Granularity is the extent to which a system is broken down into small parts, either the system itself or its description or observation. Coarse-grained systems consist of fewer, larger components than fine-grained systems; a coarse-grained description of a system regards large subcomponents while a fine-grained description regards smaller components of which the larger ones are composed. In the projects Senseval 1 and 2, fine grained and coarse grained systems are used as methods of evaluation. An optional sense hierarchy or sense grouping to allow for fine or coarse grained sense distinctions to be used in scoring. General guidelines for designing tasks were issued to ensure common evaluation standards (Edmonds 2000).Coarse grained systems to be used as evaluation methods are mentioned in the book WSD Algorithms and Applications (Agirre and Edmonds, 2006).

Table 2.5 The Senses of the Words and Their Markings

Words	**Word Senses**
BAS	1. Uygunsuz vaziyette yakalamak, duman sis vb. çevreyi kaplamak, bir yere zorla girmek 2. Bir şeyin üzerine kuvvet vererek itmek, ayak tabanını üzerine koymak 3. Çoğaltmak, basım işini gerçekleştirmek (a)
GÜL	1. İnsan, hoşuna veya tuhafına giden olaylar, durumlar karşısında, genellikle sesli bir biçimde duygusunu açığa vurmak 2. Gülgillerin örnek bitkisi, bu bitkinin katmerli, genellikle kokulu olan çiçeği. (b)
KIR	1. Sert şeyleri vurarak veya ezerek parçalamak, iri parçalara ayırmak 2. Şehir ve kasabaların dışında kalan, çoğu boş ve geniş yer, açık yer (c)
YÜZ	1. Doksan dokuzdan sonra gelen sayının adı. 2. Başta, alın, göz, burun, ağız, yanak ve çenenin bulunduğu ön bölüm, sima, çehre, surat 3. Kol, bacak, yüzgeç vb. organların özel hareketleriyle su yüzeyinde veya su içinde ilerlemek, durmak (d)

At the end of each sentence, it is indicated with a **Number**, which word sense the word was used with. Also, in some of the sentences the target word appears twice. When making the indication, (**Number, Number)** is used in that case to make indication. The first number represents the word sense of the first word, and the second number represents the second one.

For example,

Yüz çocuk denizde yüzdü. (Hundred children swam in the sea). (1,3).

In this sentence, the word yüz(hundred) in the sense of a number is listed in the word sense table as 1, and the word yüzdü (swam) is listed in the table as 3.

A sample of each sentence that was chosen regarding the words chosen is shown in Table 2.6.

Table 2.6 Sample Sentences

Words	Sentences
BAS	1. Narkotik şube son basılan evde ele geçirilen uyuşturucu miktarını açıkladı. 2. Çalıların ve ayağıma dolanan ayrık otlarının üzerine basarak yürüyordum. 3. Tarama sonuçlarını kayıt etmek veya kağıda basmak isterseniz ilk önce bilgileri kayıt etmeniz gerekmektedir.
GÜL	1. Şermin, peçeteyi ağzına götürerek birdenbire gülmeye başladı. 2. Anneme anneler gününde bir demet beyaz gül aldım.
KIR	1. Yaramaz çocuk bardak reyonunda koşarken bardaklara çarpıp kırılmalarına sebep oldu. 2. Rüzgarda savrulmak, sularda kıvrılmak, kırlarda dolaşmak ne hoştur.
YÜZ	1. İmparatorluğun yüz yılı isimli sergiye gittim. 2. Yüz bakımını sağlıklı bir şekilde yapmak cildi gençleştirir. 3. Denizde yüzerken çıkan fırtınada boğuldu.

The total number of sentences, word senses, POS and such information in the selected collection sentences are listed in Table 2.7.

Table 2.7 Total Number of Sentences, Word Senses, POS

Words	Total Number of Sentences	Number of Selected Sense	POS	
			Verb	Noun
Bas	304	3	3	-
Gül	200	2	1	1
Kır	200	2	1	1
Yüz	304	3	1	2
TOTAL	**1008**	**10**	**6**	**4**

2.2 SELECTION OF THE FEATURES EFFECTING THE WORD SENSE

The features effecting the word sense show huge diversity. Especially, it is important to decide how many of the words used with the target word will be selected. In this matter, Weaver (Weaver, 1949) has shown the role of selecting a number N of words on both sides in such a way so that the sense of the target word can be assumed. In this phase, the important question is: What shall the value of N be? By WSD, it is impossible to solve the sense as a function merely depending on the value of N. What a person has learned before is also important for WSD.

The features effecting the word sense shall be selected very carefully for the structure for WSD research to be functioning well. Features to be chosen for WSD research can be listed as follows (Mihalcea, 2002):

- The word for sense disambiguation and the type of the word.

For example,

Masadaki bardağa elim çarpınca bardak düşüp kırıldı. (As my hand hit the glass on the table, the glass fell and broke.)

In this sentence, the word "*kırıldı"* is the word for sense disambiguation. The type of this word is verb.

- Words and types of words around the target word

Masadaki (On the table)	bardağa (the glass)	elim (my hand)	çarpınca (hit)	bardak (glass)	düşüp (fell)	Kırıldı. (broke).
İsim (Noun)	İsim (Noun)	İsim (Noun)	Fiil (Verb)	İsim (Noun)	Fiil (Verb)	Fiil (Verb)

- Key words related to the sense of the word

düşmek, çarpmak (To fall, to hit)

- Word pairs

Tuz buz oldu, kırıldı.(Broken to pieces, broken).

- Local syntax

There are several features effecting the word sense, since Turkish is a language with an agglutinative structure. The type of the word and the suffix of the word have a strong influence on the WSD. But in some word types, the word before and after also plays an important role.

In our research, structural features like the suffixes of target words, the type of words that are used with them and the suffixes they get have been examined and based upon this, accuracy on correct determination of correct the sense of the sentence was identified. The features chosen for our research are as follows:

- Part Of Speech of Two Preceding Word (TPP)
- Suffix of Two Preceding Word (TPS)
- Part Of Speech of Preceding Word (PP)
- Suffix of Preceding Word (PS)
- Part Of Speech of Target Word (TP)
- Suffix of Target Word (TS)
- Part Of Speech of Succeeding Word (SP)
- Suffix of Succeeding Word (SS)
- Part Of Speech of Two Succeeding Word (TSP)
- Suffix of Two Succeeding Word (TSS)

The words, types and suffixes, that have been used after all the sentences have been analyzed, are listed in Table 2.8.

Table 2.8 The Word Types and Suffixes

Turkish POS	English POS	Turkish POS	English POS
ISIM	NOUN	FIIL	VERB
EDAT	PREPOSITION	SAYI	NUMBER
OZEL	PROPER NOUN	SIFAT	ADJECTIVE
KISALTMA	ABBREVIATURE	ZAMAN	TENSE
ZAMIR	PRONOUN	ISIM_BELIRTME_I	ACCUSATIVE CASE
BAGLAC	CONJUNCTION	ISIM_YONELME_E	DATIVE CASE
ISIM_COGUL_LER	PLURAL -S	ISIM_SAHIPLIK_O_I	POSSESIVE
FIIL_MASTAR_MEK	PREPOSITION_ TO	FIIL_DONUSUM_ME	VERB TRANSFORMATION ME
FIIL_SART_SE	VERB TRANSFORMATION	FIIL_OLDURGAN_T	TRANSITIVISED VERB
FIIL_ISTEK_E	OPTATIVE	ISIM_GIBI_CE	VERBAL NOUN
ISIM_SAHIPLIK_SEN_IN	POSSESIVE (YOUR)	ISIM_TAMLAMA_IN	GENITIVE CASE
ISIM_SAHIPLIK_BEN_IM	POSSESIVE (MY)	ISIM_TANIMLAMA_DIR	
FIIL_GECMISZAMAN_DI	PAST TENSE	FIIL_EDILGENSESLI_N	PASSIVE
FIIL_DONUSUM_EN	TRANSFORMATION en	FIIL_DONUSUM_IM	TRANSFORMATION IM
FIIL_IMSI_IP	GERUNDIAL IP	FIIL_EDILGEN_IL	PASSIVE IL
FIIL_GENISZAMAN_IR	PRESENT TENSE IR	ISIM_TAMLAMA_I	POSSESIVE CONSTRUCTION I
FIIL_SUREKLILIK_EREK	CONTINUOUS VERB	ISIM_BULUNMA_LI	NOUN OCCURENCE LI
FIIL_KISI_BEN	VERB 1st PERSON SINGULAR	ISIM_TARAFINDAN_CE	
FIIL_YETERSIZLIK_E	INADEQUACY E	FIIL_OLUMSUZLUK_ME	VERB NEGATION ME
FIIL_TANIMLAMA_ICI	VERB DEFINITION ICI	ZAMAN_BELIRTME_KI	TENSE ASSERTION KI
ISIM_ILGI_CI	NOUN RELATIVE CI	FIIL_SIMDIKIZAMAN_IYOR	VERB PRESENT TENSE IYOR
FIIL_EMIR_SIZ_IN	VERB IMPERATIVE IN	FIIL_ZORUNLULUK_MELI	VERB NECESSITY CASE MELI
FIIL_DEVAMLILIK_DIKCE	VERB CONTINIOUS DIKC	FIIL_GECMISZAMAN_MIS	VERB PAST TENSE MIS
FIIL_KISI_SIZ	VERB 2nd PERSON PLURAL	ISIM_KISI_BEN_IM	NOUN 1st PERSON SINGULAR

Turkish POS	**English POS**	**Turkish POS**	**English POS**
ISIM_DURUM_LIK	NOUN SITUATION LIK	FIIL_KISI_SEN	VERB 2nd PERSON SINGULAR
ISIM_KUCULTME_CIK	NOUN DIMINUTIVE CIK	IMEK_HIKAYE_DI	PAST TENSE DI
ISIM_SAHIPLIK_SIZ_INIZ	NOUN POSSESIVE 2nd PERSON PLURAL	ISIM_KISI_ONLAR_LER	NOUN 3rd PERSON PLURAL
FIIL_BERABERLIK_IS	VERB RECIPROCAL IS	FIIL_OLUMSUZLUK_DEN	VERB NEGATION DEN
FIIL_GELECEKZAMAN_ECEK	FUTURE	FIIL_YETENEK_EBIL	APTITUDE
FIIL_DONUSUM_IS	TRANSFORMATION VERB	FIIL_EMIR_O_SIN	VERB IMPERATIVE 3rd PERSON SINGULAR SIN
ISIM_BULUNMA_KI	NOUN OCCURENCE KI	FIIL_DONUSUM_MIS	VERB TRANSFORMATION MIS
SAYI_SIRA_INCI	NUMBER&SERIES	ISIM_CIKMA_DEN	ABLATIVE CASE
FIIL_ETTIRGEN_TIR	CAUSATIVE	ISIM_KALMA_DE	LOCATIVE CASE DE
ISIM_BIRLIKTELIK_LE	COMITATIVE NOUN LE	FIIL_BELIRTME_DIK	VERB ASSERTION DIK
ISIM_SAHIPLIK_ONLAR_LERI	NOUN POSSESSIVE 3rd PERSON PLURAL	ISIM_DONUSUM_LE	NOUN TRANSFORMATION LE
FIIL_ZAMAN_INCE		IMEK_ZAMAN_KEN	
ISIM_BULUNMA_LIK	NOUN OCCURENCE LIK	ISIM_SAHIPLIK_BIZ_IMIZ	NOUN POSSESIVE IMIZ
FIIL_EMIR_SIZRESMI_INIZ	VERB IMPERATIVE INIZ	FIIL_GIBI_CESINE	VERB GERUNDIAL CESINE
FIIL_DONUSUM_INTI	VERB TRANSFORMATION INTI	FIIL_DONUSUM_ECEK	VERB TRANSFORMATION ECEK
ISIM_YOKLUK_SIZ	PREPOSITION_IN	ISIM_DONUSUM_LES	NOUN TRANSFORMATION LES
FIIL_DONUSUM_IK	VERB TRANSFORMATION IK	ZAMIR_SAHIPLIK_IN	PRONOUN POSSESIVE IN

In the research, suffixes closer to the target word have been chosen, since these suffixes closer to the target word are more constitutive for WSD.

In Table 2.9 the appearance of these features in one sample sentence can be seen.

Table 2.9 The Word Types and Suffixes for a Sample Sentence.

Attributes	Sahnede kendimi tutamayıp gülme krizine girdim.
TPP	kendimi → tip: **İsim (Noun)**
TPS	kendimi → **İsim_Belirtme_i (Accusative Case)**
PP	tutamayıp → tip:**Fiil (Verb)**
PS	tutamayıp → **Fiil_İmsi_İp**
TP	gülme → tip:**Fiil (Verb)**
TS	**gülme → Fiil_Dönüşüm_Me (Verb Transformation Me)**
SP	krizine → tip:**İsim (Noun)**
SS	krizine → **İsim_Yönelme_e (Dative Case)**
TSP	girdim → tip:**Fiil (Verb)**
TSS	girdim → **Fiil_Kişi_Ben (1st Person Single)**

2.3 PROJECT ZEMBEREK

After features designation process, the process of separation of base types and the suffixes has been done using ZEMBEREK program.

Zemberek is an open source, platform independent, general purpose Natural Language Processing library and toolset designed for Turkic languages, especially Turkish[7]. The project is developed in java and was coded open source, thus the project is able to be installed and used in eclipse environment.

The project consists of parts like examining, morphological analysis, Ascii >Tr, Tr>Ascii transformation, syllabification and proposition. In this research, the morphological analysis feature of the project Zemberek was used. This feature separates the

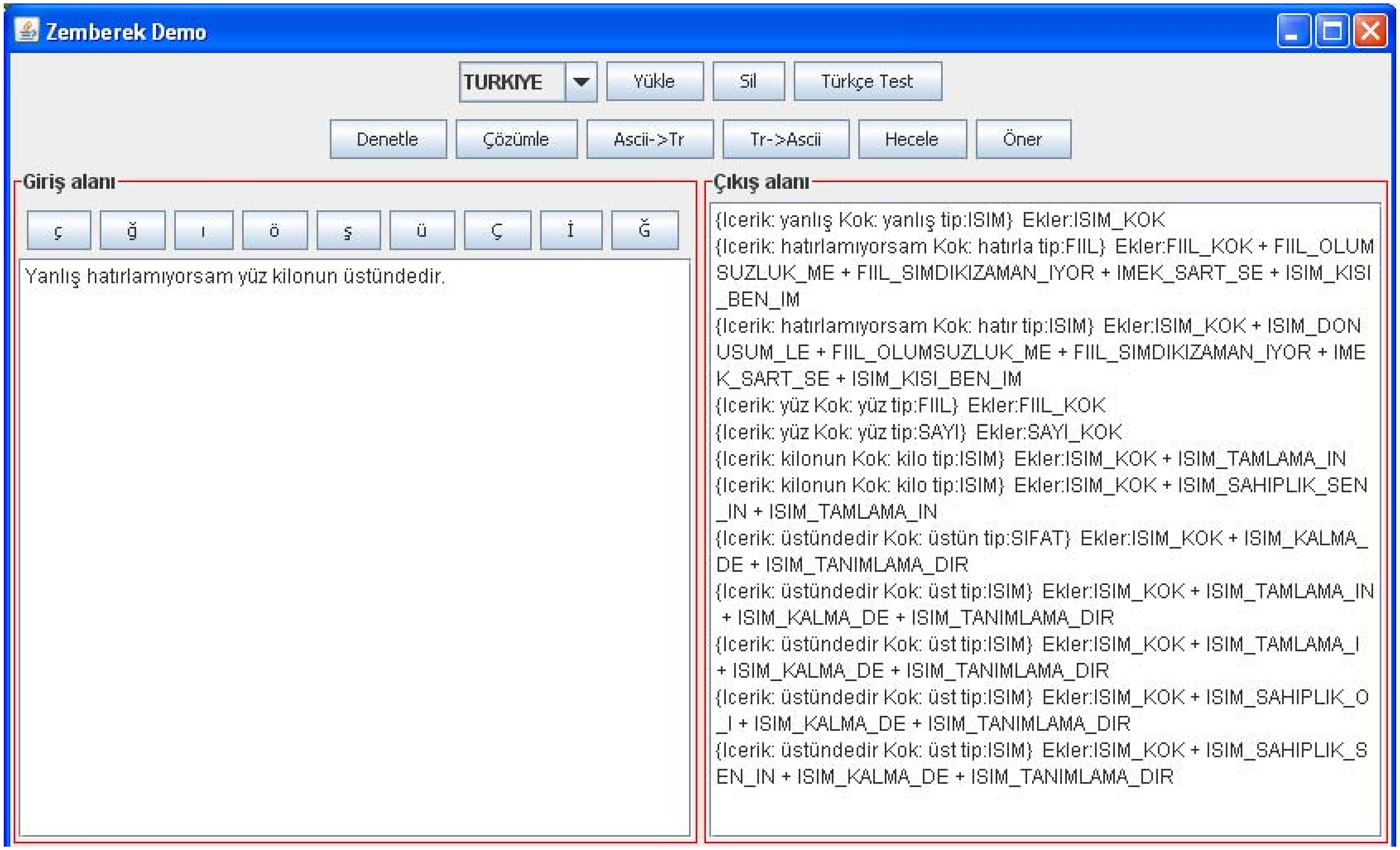
[7] Zemberek: http://code.google.com/p/zemberek/

word as base and suffix and displays their types. In Figure 2.1 you can see a morphological analysis of a sentence using Zemberek.

“ Yanlış hatırlamıyorsam yüz kilonun üstündedir. (As far as I remember, he weighs more than hundred kilos).”

The morphological analysis of the sample sentence that was written in the input field is displayed in **Table 2.10**.

When we look at the output area, it is noticed that the program separates the base and suffixes of all words. Furthermore the program offers for some words more than one alternatives. This counts for both the target word and the neighboring words. For the system to be conceived, the features that were chosen for each sentence, along with the morphological analysis process, have to be decoded properly. Thus, for the values in the output area, which appear more than once, to be properly decoded, each of the output area values have to be examined one by one and the correct morphological analysis has to be manually selected.



Yanlış hatırlamıyorsam yüz kilonun üstündedir.

```
{Icerik: yanlış Kok: yanlış tip:ISIM}  Ekler:ISIM_KOK
{Icerik: hatırlamıyorsam Kok: hatırla tip:FIIL}  Ekler:FIIL_KOK + FIIL_OLUMSUZLUK_ME + FIIL_SIMDIKIZAMAN_IYOR + IMEK_SART_SE + ISIM_KISI_BEN_IM
{Icerik: hatırlamıyorsam Kok: hatır tip:ISIM}  Ekler:ISIM_KOK + ISIM_DONUSUM_LE + FIIL_OLUMSUZLUK_ME + FIIL_SIMDIKIZAMAN_IYOR + IMEK_SART_SE + ISIM_KISI_BEN_IM
{Icerik: yüz Kok: yüz tip:FIIL}  Ekler:FIIL_KOK
{Icerik: yüz Kok: yüz tip:SAYI}  Ekler:SAYI_KOK
{Icerik: kilonun Kok: kilo tip:ISIM}  Ekler:ISIM_KOK + ISIM_TAMLAMA_IN
{Icerik: kilonun Kok: kilo tip:ISIM}  Ekler:ISIM_KOK + ISIM_SAHIPLIK_SEN_IN + ISIM_TAMLAMA_IN
{Icerik: üstündedir Kok: üstün tip:SIFAT}  Ekler:ISIM_KOK + ISIM_KALMA_DE + ISIM_TANIMLAMA_DIR
{Icerik: üstündedir Kok: üst tip:ISIM}  Ekler:ISIM_KOK + ISIM_TAMLAMA_IN + ISIM_KALMA_DE + ISIM_TANIMLAMA_DIR
{Icerik: üstündedir Kok: üst tip:ISIM}  Ekler:ISIM_KOK + ISIM_TAMLAMA_I + ISIM_KALMA_DE + ISIM_TANIMLAMA_DIR
{Icerik: üstündedir Kok: üst tip:ISIM}  Ekler:ISIM_KOK + ISIM_SAHIPLIK_O_I + ISIM_KALMA_DE + ISIM_TANIMLAMA_DIR
{Icerik: üstündedir Kok: üst tip:ISIM}  Ekler:ISIM_KOK + ISIM_SAHIPLIK_SEN_IN + ISIM_KALMA_DE + ISIM_TANIMLAMA_DIR
```

Figure 2.1 Zemberek Screen Shot

Table 2.10 Zemberek Output for a Sample Sentence

Root of The Word	Type of the Word	1. Affix	2. Affix	3. Affix	4. Affix	5. Affix
Icerik: yanlış Kok: yanlış	tip:ISIM					
Icerik: hatırlamıyorsam Kok: hatırla	tip:FIIL	FIIL_OLUMSUZLUK_ME	FIIL_SIMDIKIZAMAN_IYOR	IMEK_SART_SE	ISIM_KISI_BEN_IM	
Icerik: hatırlamıyorsam Kok: hatır	tip:ISIM	ISIM_DONUSUM_LE	FIIL_OLUMSUZLUK_ME	FIIL_SIMDIKIZAMAN_IYOR	IMEK_SART_SE	ISIM_KISI_BEN_IM
Icerik:yüz Kok: yüz	tip:FIIL					
Icerik:yüz Kok: yüz	tip:SAYI					
Icerik:kilonun Kok: kilo	tip:ISIM	ISIM_TAMLAMA_IN				
Icerik:kilonun Kok: kilo	tip:ISIM	ISIM_SAHIPLIK_SEN_IN	ISIM_TAMLAMA_IN			
Icerik:üstündedir Kok: üstün	tip:SIFAT	ISIM_KALMA_DE	ISIM_TANIMLAMA_DIR			
Icerik:üstündedir Kok: üst	tip:ISIM	ISIM_TAMLAMA_IN	ISIM_KALMA_DE	ISIM_TANIMLAMA_DIR		
Icerik:üstündedir Kok: üst	tip:ISIM	ISIM_TAMLAMA_I	ISIM_KALMA_DE	ISIM_TANIMLAMA_DIR		
Icerik:üstündedir Kok: üst	tip:ISIM	ISIM_SAHIPLIK_O_I	ISIM_KALMA_DE	ISIM_TANIMLAMA_DIR		
Icerik:üstündedir Kok: üst	tip:ISIM	ISIM_SAHIPLIK_SEN_IN	ISIM_KALMA_DE	ISIM_TANIMLAMA_DIR		

2.4 ZEMBEREK PARSER PROGRAM

In the next phase, a program called Zemberek Parser was written. This program was written in JAVA language. The purpose of this program is the classification of the output values found with the morphological analysis process in a file, and to enable the user for the selection of the correct morphological analysis manually, if there are more than one alternative. Thus, a feature called ZEMBEREK library was added to the program. In Figure 2.2 you can see the interface of the program.

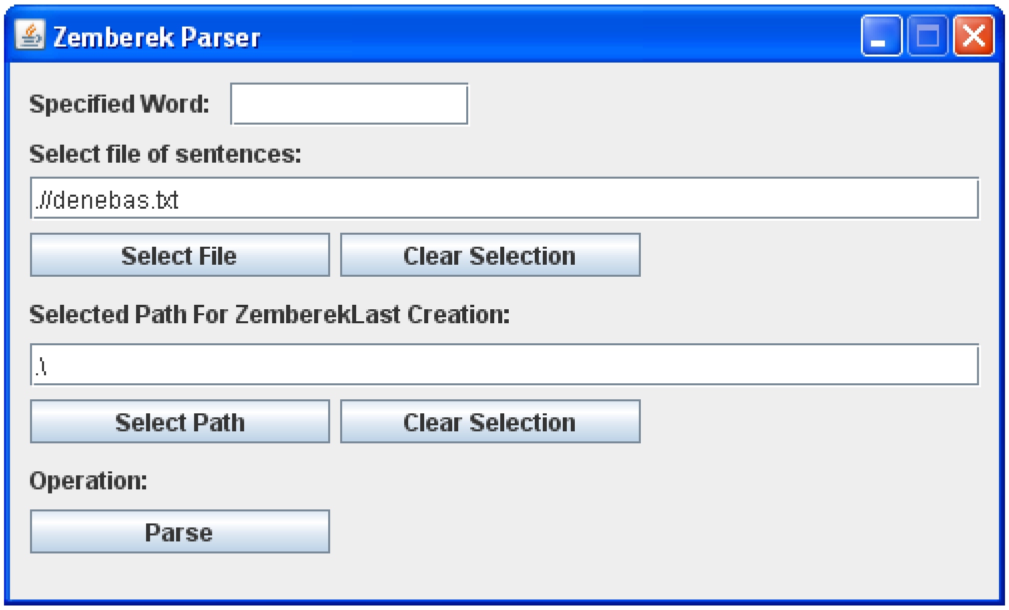
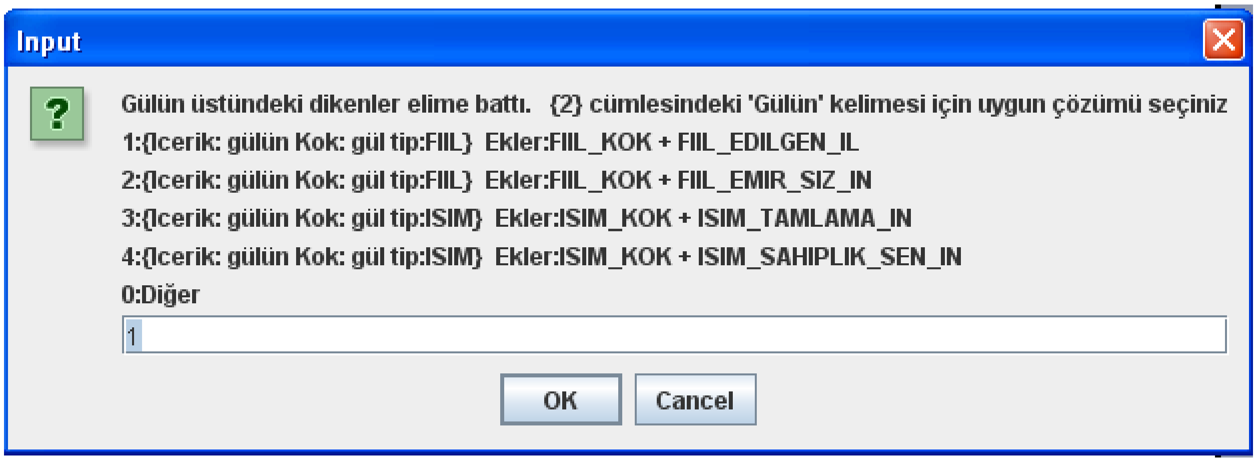


Figure 2.2 Zemberek Parser Screen Shot

First, the target word was written in the program, then the txt files with sentences including the target words were selected, and then the parsing process was done. In this process, the morphological analysis feature of the program ZEMBEREK was used, to be able to do morphological analysis all of the sentences. Since the program Zemberek offered more than one morphological analysis for some words, it was important to choose the most appropriate morphological analysis for the senses and types of the word. Therefore, the interface enabling the user to choose the correct morphological analysis manually was written. In the screenshot, you can see the interface of the Zemberek Parser.

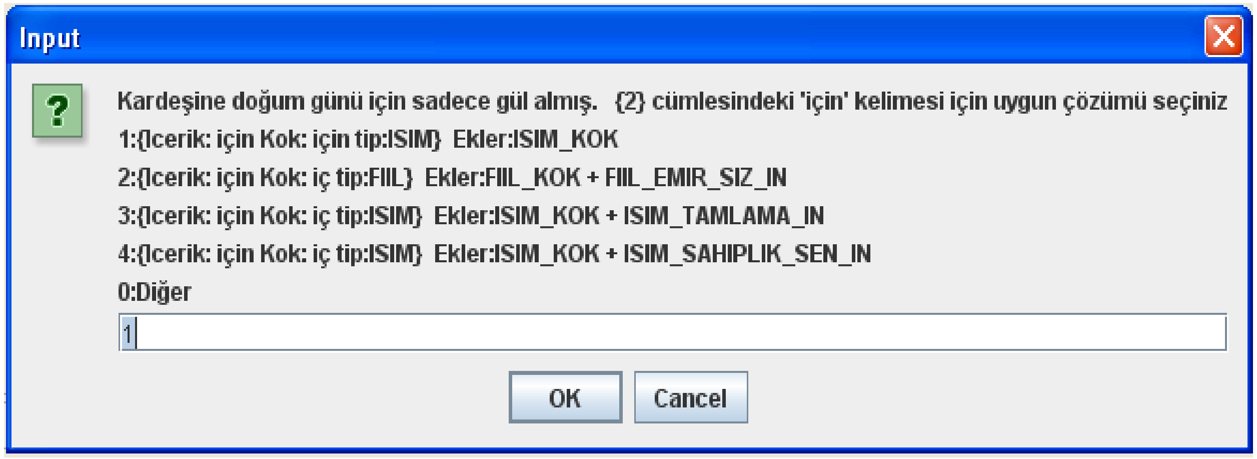


Gülün üstündeki dikenler elime battı. {2} cümlesindeki 'Gülün' kelimesi için uygun çözümü seçiniz
1:{Icerik: gülün Kok: gül tip:FIIL} Ekler:FIIL_KOK + FIIL_EDILGEN_IL
2:{Icerik: gülün Kok: gül tip:FIIL} Ekler:FIIL_KOK + FIIL_EMIR_SIZ_IN
3:{Icerik: gülün Kok: gül tip:ISIM} Ekler:ISIM_KOK + ISIM_TAMLAMA_IN
4:{Icerik: gülün Kok: gül tip:ISIM} Ekler:ISIM_KOK + ISIM_SAHIPLIK_SEN_IN
0:Diğer
1

Figure 2.3 Zemberek Parser Input Screen for the Word "Gül"

It is seen in Figure 2.3, the program Zemberek offers 4 options for the target word. Two of them have verbs as bases, and two of them nouns. Still, there are two options for the suffix, depending on the type of the word. (Verb or noun).

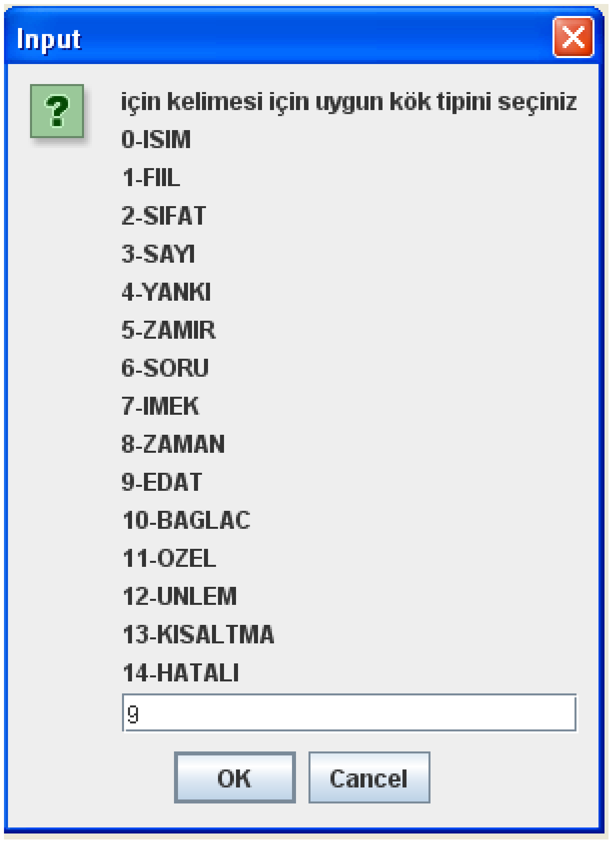
But for some words, Zemberek can not offer a correct morphological analysis. For this reason, an option called "0 other" was also included, which enables the user to choose manually, when there are no correct morphological analyses in the morphological analysis results list. An example for this is shown in Figure 2.4.



Kardeşine doğum günü için sadece gül almış. {2} cümlesindeki 'için' kelimesi için uygun çözümü seçiniz
1:{Icerik: için Kok: için tip:ISIM} Ekler:ISIM_KOK
2:{Icerik: için Kok: iç tip:FIIL} Ekler:FIIL_KOK + FIIL_EMIR_SIZ_IN
3:{Icerik: için Kok: iç tip:ISIM} Ekler:ISIM_KOK + ISIM_TAMLAMA_IN
4:{Icerik: için Kok: iç tip:ISIM} Ekler:ISIM_KOK + ISIM_SAHIPLIK_SEN_IN
0:Diğer
1

Figure 2.4 Zemberek Parser Input screen for the Word "İçin"

The word "İçin" in Turkish indicates a cause and effect relation, and therefore it is classified as a preposition in all four morphological analysiss. So, using the 0: other option, it is possible to mark the word "*için*" as preposition. It is shown in Figure 2.5.



için kelimesi için uygun kök tipini seçiniz
0-ISIM
1-FIIL
2-SIFAT
3-SAYI
4-YANKI
5-ZAMIR
6-SORU
7-IMEK
8-ZAMAN
9-EDAT
10-BAGLAC
11-OZEL
12-UNLEM
13-KISALTMA
14-HATALI
9

Figure 2.5 The Zemberek Parser Input screen for the "diğer" Option

If the word has a suffix, it is possible to choose a suffix type in another area after the indication.

Using the Input interface of the Zemberek Parser program, selections were made starting from two preceding words' type and suffix until two succeeding words' type and suffix. Since our data consist of 1000 sample sentences and selections regarding word type and suffix of 5 words between the range from word type and suffix of 2 words before to word type and suffix of 2 words after is being done, it is a time consuming process to save the data in appropriate format. This operation has to be done manually, since it can not be done automatically. The results acquired from this were all saved in a file with arff extension. An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes[8]. Thus, the sentences selected were converted to the structure to be used in the project

[8] ARFF: file format http://www.cs.waikato.ac.nz/~ml/weka/arff.html

Zemberek Parser Pseudocode

Table 2.11 Main function of Zemberek Parser Pseudocode

```
Function main
 Declare word-anlamı aranan kelime-, initial value from GUI
 Declare file contains sentences, each sentences in new line.
 Declare dataList contains attributes of each data
  For each sentences in file
   Tokenize sentences according to space (\t, \r, \s+); last token contains sense of words.
  For each token in sentence
    Send zemberek for morphological analysis, it returns solutions set
    For root of each solutions check.
        If there is at least one root equal word, set attributes for this token
            Send this token to SetWordDetails function
            If there is one token before this token
                  Send preceding token to SetWordDetails function
                  If there are two token before this token
                     Send two preceding token to SetWordDetails function
                  If there is one token after this token
                     Send succeeding token to SetWordDetails function
                  If there are two token after this token
                     Send two succeeding after token to SetWordDetails function
                     Extract word sense from last token
                     Add dataList with all attributes of this token
        Else
            Quit solution loop
 // create arff file
 For each attribute type (part of speech of two preceding word, suffix of succeeding word, etc)
    Select different types from dataList
    Save attributes part of arrf file
    Save data part of arff file.
```

Table 2.12 Sub-function of Zemberek Parser Pseudocode

```
Function SetWordDetails (token)
    Send zemberek for morphological analysis, it returns solutions set
     If solution set is null
          Set token root type NULL
          Set token affix NULL
     Else- there is at least one solution
      If there is only one solution
          Set token root type from solution
          Set token affix from solution
      Else - there are more than one solution
          Show all solutions and other choice to user
          If user selected best solution in list
            Set token root type from best solution
            Set token affix from best solution
          Else – user selected other
            Show to user all root type options (NOUN, VERB etc) and other option
              If user selected type
                        Set token root type selected type
                Else
                     Set token root type from input box user enter type
            Show to user all affix type options (PLURAL -S,  NOUN DIMINUTIVE CIK etc)
              If user selected type
                     Set token affix type selected type
              Else
                     Set token affix type from input box user enter type
```

Table 2.13 Output as Arff Files

@relation meanings-of-words
@attribute iki-önceki-kelime-tipi {SIFAT,OZEL,ZAMAN,FIIL,NULL,ISIM,EDAT,ZAMIR}
@attribute iki-önceki-kelime-eki {NULL,ISIM_KALMA_DE,FIIL_DONUSUM_IK,FIIL_DONUSUM_EN,FIIL_GECMISZAMAN_DI,ISIM_SAHIPLIK_BIZ_IMIZ,ISIM_SAHIPLIK_SEN_IN, ISIM_CIKMA_DEN,ISIM_YOKLUK_SIZ,FIIL_ETTIRGEN_TIR,ISIM_SAHIPLIK_BEN_IM,FIIL_IMSI_IP,ISIM_TAMLAMA_IN,FIIL_EDILGEN_IL, ISIM_BULUNMA_LIK,FIIL_BERABERLIK_IS,ISIM_TAMLAMA_I,FIIL_EDILGENSESLI_N,FIIL_DONUSUM_ME,ISIM_BELIRTME_I,ISIM_COGUL_LER, FIIL_OLUMSUZLUK_ME,ISIM_YONELME_E,FIIL_BELIRTME_DIK,ISIM_TANIMLAMA_DIR}
@attribute önceki-kelime-tipi {ISIM,FIIL,EDAT,NULL,SIFAT,OZEL,ZAMIR,ZAMAN}
@attribute önceki-kelime-eki {ISIM_TAMLAMA_I,FIIL_BELIRTME_DIK,ISIM_BELIRTME_I,ISIM_BULUNMA_LI,ISIM_KUCULTME_CIK,ISIM_TAMLAMA_IN,FIIL_ZAMAN_INCE,FIIL_IMSI_IP,FIIL_SUREKLILIK_EREK,ISIM_BIRLIKTELIK_LE,ISIM_DURUM_LIK,FIIL_DONUSUM_IK, ISIM_ILISKILI_SEL,FIIL_EDILGENSESLI_N,NULL,ISIM_CIKMA_DEN,ISIM_KALMA_DE,FIIL_OLUMSUZLUK_ME,FIIL_DONUSUM_IM, ISIM_YONELME_E,FIIL_DONUSUM_EN,FIIL_GENISZAMAN_IR,ISIM_SAHIPLIK_BEN_IM,FIIL_EDILGEN_IL,ISIM_SAHIPLIK_BIZ_IMIZ, ISIM_BULUNMA_LIK,FIIL_DONUSUM_ME,ISIM_SAHIPLIK_SEN_IN,ISIM_ILGI_CI,ISIM_COGUL_LER}
@attribute anlami-aranan-kelime-tipi {FIIL,ISIM}
@attribute anlami-aranan-kelime-eki {NULL,FIIL_GECMISZAMAN_DI,FIIL_ETTIRGEN_TIR,FIIL_GENISZAMAN_IR,FIIL_SUREKLILIK_EREK, FIIL_SIMDIKIZAMAN_IYOR,FIIL_BELIRTME_DIK,FIIL_DONUSUM_ME,FIIL_MASTAR_MEK,FIIL_GECMISZAMAN_MIS,FIIL_IMSI_IP, FIIL_ISTEK_E,ISIM_BELIRTME_I,ISIM_YONELME_E,ISIM_BULUNMA_LI,ISIM_TAMLAMA_I,ISIM_TAMLAMA_IN,ISIM_SAHIPLIK_O_I, ISIM_SAHIPLIK_SEN_IN,ISIM_SAHIPLIK_BEN_IM,ISIM_KALMA_DE,ISIM_COGUL_LER,ISIM_BIRLIKTELIK_LE,ISIM_SAHIPLIK_SIZ_INIZ, ISIM_SAHIPLIK_BIZ_IMIZ}
@attribute sonraki-kelime-tipi {ISIM,FIIL,NULL,ZAMIR,BAGLAC,EDAT,OZEL,SIFAT,ZAMAN}
@attribute sonraki-kelime-eki

{NULL,ISIM_SAHIPLIK_SEN_IN,ISIM_KALMA_DE,ISIM_SAHIPLIK_BIZ_IMIZ,FIIL_DONUSUM_EN,FIIL_GECMISZAMAN_DI,FIIL_YETERSIZLIK_E,FIIL_SUREKLILIK_EREK,FIIL_ZAMAN_INCE,FIIL_DONUSUM_ME,FIIL_SIMDIKIZAMAN_IYOR,FIIL_MASTAR_MEK,ISIM_DONUSUM_LE,FIIL_ETTIRGEN_TIR,ISIM_DONUSUM_LES,FIIL_OLDURGAN_T,FIIL_EMIR_SIZ_IN,FIIL_EDILGEN_IL,ISIM_SAHIPLIK_O_I,FIIL_EDILGENSESLI_N,FIIL_OLUMSUZLUK_ME,FIIL_GENISZAMAN_IR,FIIL_BELIRTME_DIK,ISIM_TAMLAMA_I,ISIM_BULUNMA_LI,ISIM_TAMLAMA_IN,FIIL_GECMISZAMAN_MIS,ISIM_DURUM_LIK,ISIM_CIKMA_DEN,ISIM_ILGI_CI,ISIM_COGUL_LER,ISIM_SAHIPLIK_BEN_IM,ISIM_YONELME_E,ISIM_BULUNMA_LIK,ISIM_BELIRTME_I}
@attribute iki-sonraki-kelime-tipi {ISIM,NULL,FIIL,EDAT,ZAMIR,SIFAT,ZAMAN,BAGLAC}
@attribute iki-sonraki-kelime-eki {NULL,ISIM_TAMLAMA_I,ISIM_YONELME_E,ISIM_BULUNMA_LI,FIIL_ETTIRGEN_TIR,FIIL_EMIR_O_SIN,ISIM_SAHIPLIK_BIZ_IMIZ,FIIL_YETERSIZLIK_E,FIIL_DONUSUM_IM,IMEK_HIKAYE_DI,SAYI_SIRA_INCI,FIIL_SUREKLILIK_EREK,ISIM_DURUM_LIK,FIIL_GECMISZAMAN_DI,ISIM_BELIRTME_I,ISIM_KALMA_DE,ISIM_YOKLUK_SIZ,ISIM_BIRLIKTELIK_LE,ISIM_SAHIPLIK_BEN_IM,ISIM_DONUSUM_LE,ISIM_DONUSUM_LES,ISIM_SAHIPLIK_SEN_IN,FIIL_SART_SE,FIIL_MASTAR_MEK,FIIL_EDILGENSESLI_N,FIIL_DONUSUM_ME,ISIM_COGUL_LER,FIIL_EDILGEN_IL,ISIM_SAHIPLIK_O_I,ISIM_TAMLAMA_IN,FIIL_GELECEKZAMAN_ECEK,FIIL_EMIR_SIZ_IN,FIIL_SIMDIKIZAMAN_IYOR,ISIM_SAHIPLIK_SIZ_INIZ,FIIL_GECMISZAMAN_MIS,ZAMAN_BELIRTME_KI,FIIL_GENISZAMAN_IR,FIIL_DONUSUM_EN,ISIM_CIKMA_DEN,FIIL_OLUMSUZLUK_ME,ISIM_KUCULTME_CIK}
@attribute anlam {1,2,3}

Table 2.14 A Sample of Arff Files

```
@data
 %
 % 304 instance(s)
 %
 OZEL,ISIM_KALMA_DE, ISIM,NULL,  ISIM,NULL,  ISIM,NULL,  ISIM,NULL, 1
 ZAMAN,NULL, ISIM,ISIM_TAMLAMA_I,  ISIM,NULL,  ISIM,NULL,  NULL,NULL, 1
 FIIL,FIIL_DONUSUM_EN, EDAT,NULL,  ISIM,NULL,  ISIM,NULL,  ISIM,NULL, 1
 NULL,NULL, NULL,NULL,  ISIM,NULL,  ISIM,NULL,  ISIM,ISIM_YONELME_E, 1
 NULL,NULL, NULL,NULL,  ISIM,NULL,  ISIM,NULL,  ISIM,NULL, 1
 NULL,NULL, NULL,NULL,  ISIM,NULL,  ISIM,NULL,  ISIM,ISIM_BULUNMA_LI, 1
 ISIM,NULL, SIFAT,NULL,  ISIM,NULL,  ISIM,ISIM_SAHIPLIK_SEN_IN,  FIIL,FIIL_ETTIRGEN_TIR, 1
 SIFAT,NULL, ISIM,ISIM_TAMLAMA_I,  ISIM,NULL,  FIIL,FIIL_SUREKLILIK_EREK,  ZAMIR,NULL, 1
ISIM,NULL, ISIM,ISIM_KALMA_DE,  FIIL,FIIL_GECMISZAMAN_DI,  NULL,NULL,  NULL,NULL, 3
 NULL,NULL, NULL,NULL,  ISIM,NULL,  ISIM,NULL,  ISIM,ISIM_SAHIPLIK_O_I, 2
 EDAT,NULL, FIIL,FIIL_EDILGEN_IL,  ISIM,NULL,  ISIM,ISIM_COGUL_LER,  FIIL,FIIL_GECMISZAMAN_DI, 2
 ISIM,NULL, FIIL,FIIL_DONUSUM_EN,  ISIM,NULL,  ISIM,ISIM_TAMLAMA_I,  FIIL,FIIL_GECMISZAMAN_DI, 2
 NULL,NULL, NULL,NULL,  ISIM,NULL,  FIIL,FIIL_DONUSUM_ME,  ISIM,ISIM_TAMLAMA_I, 2
 NULL,NULL, SIFAT,NULL,  ISIM,NULL,  ISIM,ISIM_COGUL_LER,  ISIM,NULL, 2
NULL,NULL, ISIM,ISIM_TAMLAMA_IN,  ISIM,ISIM_BELIRTME_I,  SIFAT,NULL,  FIIL,FIIL_EDILGENSESLI_N, 2
 SIFAT,NULL, ISIM,NULL,  ISIM,NULL,  SIFAT,NULL,  SIFAT,NULL, 2
 NULL,NULL, NULL,NULL,  ISIM,NULL,  FIIL,FIIL_DONUSUM_ME,  ISIM,ISIM_SAHIPLIK_O_I, 2
 ISIM,NULL, ISIM,ISIM_COGUL_LER,  FIIL,FIIL_BELIRTME_DIK,  FIIL,FIIL_GECMISZAMAN_DI,  NULL,NULL, 3
 NULL,NULL, ISIM,ISIM_KALMA_DE,  FIIL,FIIL_GECMISZAMAN_DI,  ZAMAN,NULL,  ISIM,NULL, 3
 ZAMIR,ISIM_YONELME_E, SIFAT,NULL,  FIIL,FIIL_SIMDIKIZAMAN_IYOR,  NULL,NULL,  NULL,NULL, 3
 ISIM,NULL, ISIM,ISIM_BIRLIKTELIK_LE,  FIIL,FIIL_BELIRTME_DIK,  FIIL,FIIL_EDILGEN_IL,  NULL,NULL, 3
 ISIM,ISIM_YOKLUK_SIZ, ISIM,ISIM_KALMA_DE,  FIIL,FIIL_SIMDIKIZAMAN_IYOR,  NULL,NULL,  NULL,NULL, 3
 ISIM,NULL, ZAMAN,NULL,  FIIL,FIIL_DONUSUM_ME,  ISIM,ISIM_DONUSUM_LE,  NULL,NULL, 3
 ISIM,NULL, ISIM,ISIM_KUCULTME_CIK,  FIIL,FIIL_BELIRTME_DIK,  FIIL,FIIL_EDILGEN_IL,  NULL,NULL, 3
```

In Table 2.13 and Table 2.14 some part of the arff file output produced for “yüz” can be seen. As it is seen in Table 2.13 primarily the definitions of the attributes chosen as features and the values these attributes get are seen. First attribute was two preceding words’ type. Which types this word can take for the word “yüz” is listed. The second attribute was the suffix of two preceding word of the target word. In this list, it can be seen that suffixes of the two preceding word of the target word for 304 sentences. There are 10 such attributes. The last attribute is the word sense classification attribute.
When Table 2.14 is examined the part of the arff produced for word where “yüz” analysis for each sentence is found can be seen. In this table one can find 19 attributes related to the type and suffix of five words mentioned above and word sense classification number of the sentences.

2.5 WEKA PROJECT

WEKA(Witten and Frank, 2000) is a collection of ML algorithms for data mining tasks that can either be applied directly to a dataset or called from your own Java code[9].

Since WEKA is open sourced and written in Java language, this enables it to be easily used in other projects and to develop new algorithms. In Figure 2.6, you can see the interface of the program.

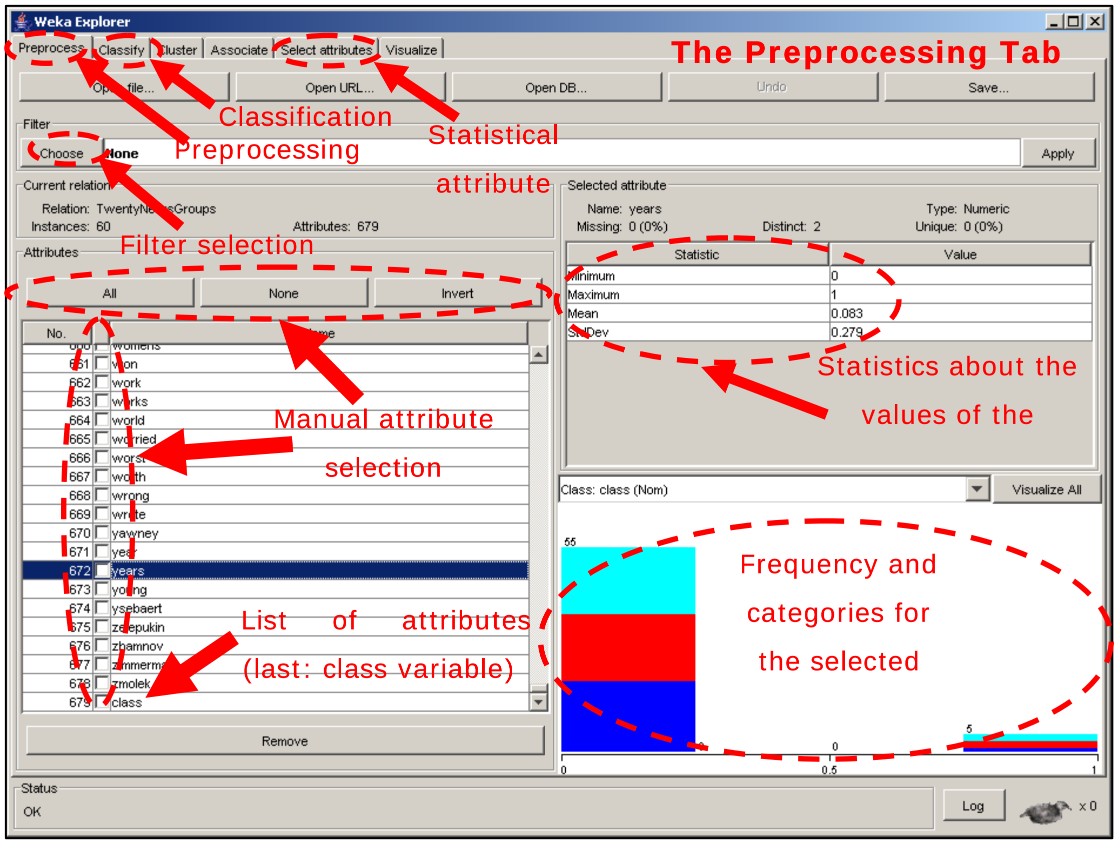
2.5.1 The WEKA Explorer

Section Tabs

- Preprocess: Choose and modify the data being acted on.
- Classify: Train and test learning schemes that classify or perform regression.
- Cluster: Learn clusters for the data.
- Associate: Learn association rules for the data.
- Select attributes: Select the most relevant attributes in the data.
- Visualize: View an interactive 2D plot of the data.

[9] WEKA http://www.cs.waikato.ac.nz/~ml/WEKA/index.html

At the top of the classify section is the Classifier box. Figure 2.7 shows this interface. One can select many classification algorithms with that button. The list of algorithms used within the project WEKA is shown in Figure 2.8.



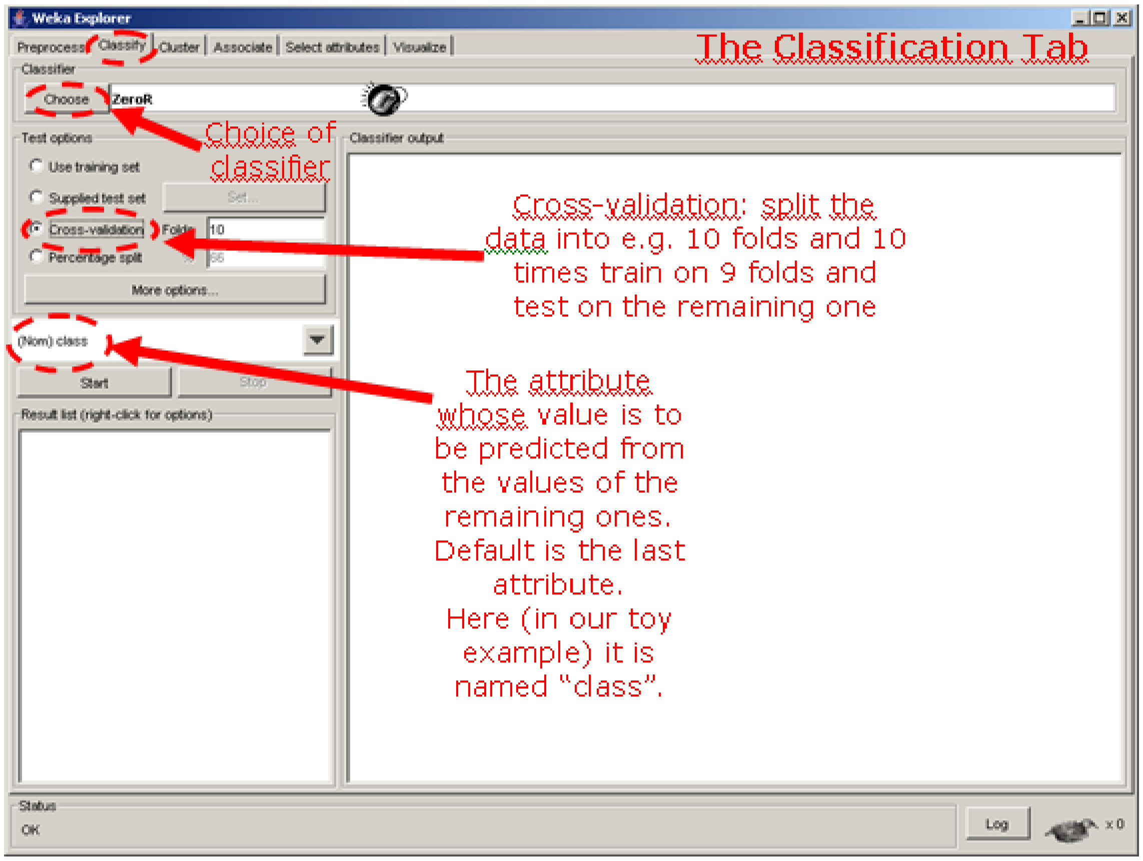
Relation: TwentyNewsGroups
Instances: 60
Attributes: 679

Name: years
Type: Numeric
Missing: 0 (0%)
Distinct: 2
Unique: 0 (0%)

Statistic	Value
Minimum	0
Maximum	1
Mean	0.083
StdDev	0.279

No.	Name
660	womens
661	won
662	work
663	works
664	world
665	worried
666	worst
667	worth
668	wrong
669	wrote
670	yawney
671	year
672	years
673	young
674	ysebaert
675	zelepukin
676	zhamnov
677	zimmerm
678	zmolek
679	class

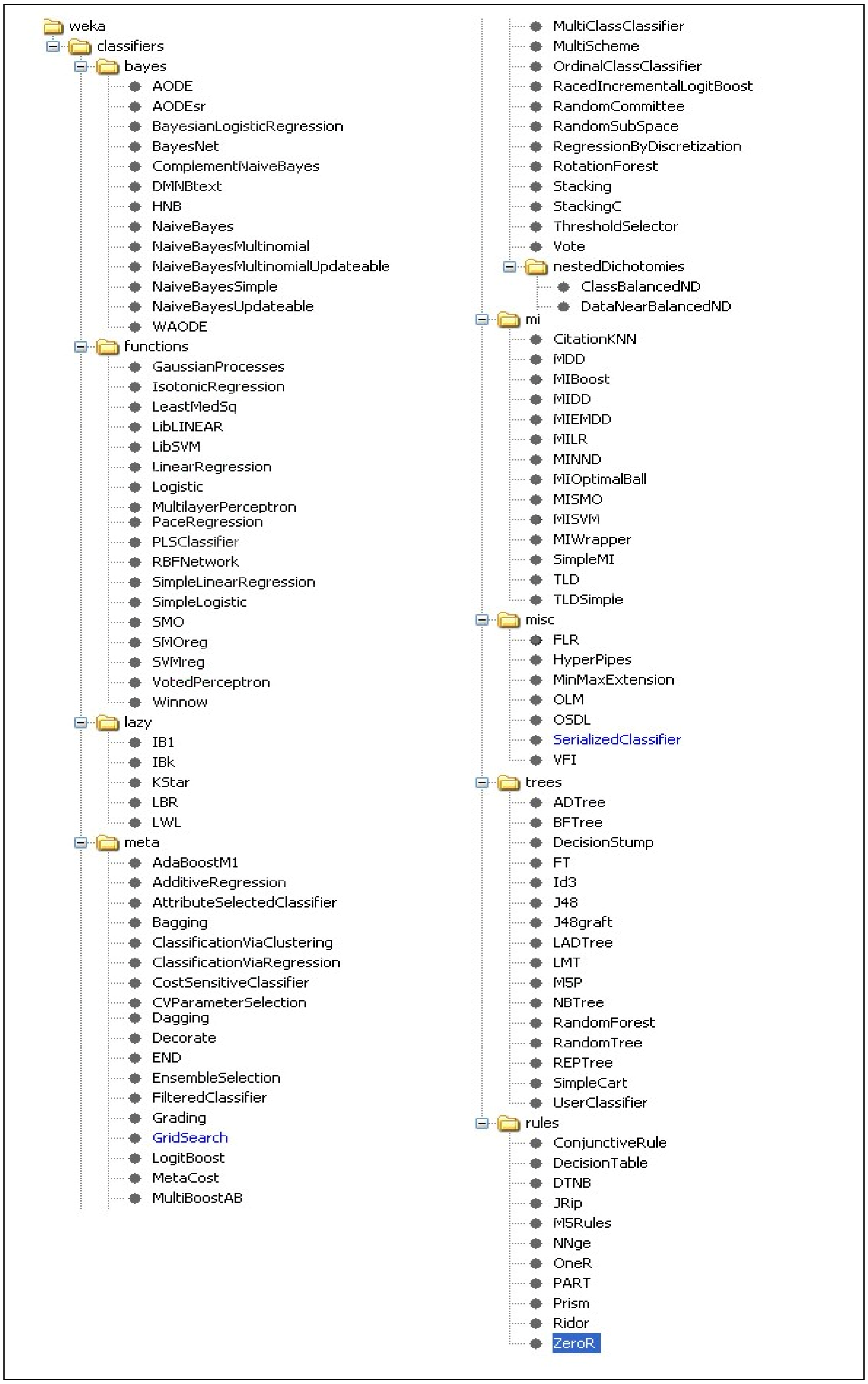
Figure 2.6 Interface of WEKA



Cross-validation: split the data into e.g. 10 folds and 10 times train on 9 folds and test on the remaining one

The attribute whose value is to be predicted from the values of the remaining ones. Default is the last attribute. Here (in our toy example) it is named "class".

Figure 2.7 WEKA Clasification Tab



```
weka
  classifiers
    bayes
      AODE
      AODEsr
      BayesianLogisticRegression
      BayesNet
      ComplementNaiveBayes
      DMNBtext
      HNB
      NaiveBayes
      NaiveBayesMultinomial
      NaiveBayesMultinomialUpdateable
      NaiveBayesSimple
      NaiveBayesUpdateable
      WAODE
    functions
      GaussianProcesses
      IsotonicRegression
      LeastMedSq
      LibLINEAR
      LibSVM
      LinearRegression
      Logistic
      MultilayerPerceptron
      PaceRegression
      PLSClassifier
      RBFNetwork
      SimpleLinearRegression
      SimpleLogistic
      SMO
      SMOreg
      SVMreg
      VotedPerceptron
      Winnow
    lazy
      IB1
      IBk
      KStar
      LBR
      LWL
    meta
      AdaBoostM1
      AdditiveRegression
      AttributeSelectedClassifier
      Bagging
      ClassificationViaClustering
      ClassificationViaRegression
      CostSensitiveClassifier
      CVParameterSelection
      Dagging
      Decorate
      END
      EnsembleSelection
      FilteredClassifier
      Grading
      GridSearch
      LogitBoost
      MetaCost
      MultiBoostAB
      MultiClassClassifier
      MultiScheme
      OrdinalClassClassifier
      RacedIncrementalLogitBoost
      RandomCommittee
      RandomSubSpace
      RegressionByDiscretization
      RotationForest
      Stacking
      StackingC
      ThresholdSelector
      Vote
      nestedDichotomies
        ClassBalancedND
        DataNearBalancedND
    mi
      CitationKNN
      MDD
      MIBoost
      MIDD
      MIEMDD
      MILR
      MINND
      MIOptimalBall
      MISMO
      MISVM
      MIWrapper
      SimpleMI
      TLD
      TLDSimple
    misc
      FLR
      HyperPipes
      MinMaxExtension
      OLM
      OSDL
      SerializedClassifier
      VFI
    trees
      ADTree
      BFTree
      DecisionStump
      FT
      Id3
      J48
      J48graft
      LADTree
      LMT
      M5P
      NBTree
      RandomForest
      RandomTree
      REPTree
      SimpleCart
      UserClassifier
    rules
      ConjunctiveRule
      DecisionTable
      DTNB
      JRip
      M5Rules
      NNge
      OneR
      PART
      Prism
      Ridor
      ZeroR
```

Figure 2.8 The List of Algorithms in WEKA

2.5.2 Methods Used In WEKA Project

The methods used for WSD were mentioned in the second chapter. In this section, it will be mentioned which classification algorithms will be used and the features of these algorithms using the program WEKA.

The classification algorithms in WEKA are shown in Figure 2.8. In this project, the NaiveBayes (NB) algorithm of Bayes group, SimpleCART algorithm of trees group, KStar algorithm of lazy group and bagging algorithm of meta group were selected.

2.5.2.1 Statistical Methods

The statistical methods based on the application of classification models and the statistical analyses on the data are presented with the title Bayes classification at WEKA. The most common method in WSD research is the NB method based upon Bayes method.

NB classifiers are based on Bayes theorem and analyze the relation between target variable and the independent variables, when examining the possibility of occurrence of events. The NB method is designed for use supervised learning tasks. In NB approach, the wider area around the word to be disambiguated for its word sense shall be examined. No features will be selected and the combination of all information is used. The statistical statement of the NB method is given as follows:

$$P(m_k|n) = \frac{P(n|m_k)}{P(n)} P(m_k) \tag{2.1}$$

In this statement:

- $m_1, m_2, \ldots m_k$ are the word senses of the words to be disambiguated.
- $n_1, n_2, \ldots n_k$ are the sentences around the word to be disambiguated.
- $f_1, f_2, \ldots f_k$ are words used as features for the words selected for WSD.

It is assumed, that the features selected in NB are independent from each other. In this case, the formula is:

$$P(n|m_k) = P(\{f_j|f_j \in n\}|m_k) = \prod_{f_j \in n} P(f_j|m_k) \quad (2.2)$$

In NB list of options are shown in Table 2.15.

Table 2.15 List of Options in NaiveBayes

Parameter	Definition
-K	Use kernel estimation for modelling numeric attributes rather than a single normal distribution.
-D	Use supervised discretization to process numeric attributes.
-O	Display model in old format (good when there are many classes)

2.5.2.2 Decision Trees:

Decision tree is an expectation technique in the form of a tree. With its tree structure, both classification processes and rules that are easy to understand can be built. The decision trees are applied in sub sections of NLP, like text classification (Weiss et. al., 1999), WSD (Brown et. al., 1991) element finding (Marquez, 1999), machine translation (Tanka, 1996) and morphological analysis (Haruno et. al., 1998).

Decision tree consist of decision nodes, branches and leaves. The decision node indicates the test to be realized. The result of this test causes the branching of the tree without losing any data. In each node, test and branching processes occur consecutively. Each branch of the tree is a candidate for accomplishing the classification process. If no classification process can be done at the tip of branch, a decision node occurs as a result of each cycle. But if the cycle constitutes a specific class in the end, there is a leaf at the tip of the branch. This leaf is one of the classes to be determined on data. The decision tree process starts at the base node and follows consecutive nodes until it reaches the leaf.

The first decision tree algorithm developed is ID3, developed by Ross Quinlan (Quinlan, 1986). The decision tree algorithm used for WSD researches are C4.5 and CART (Classification and Regression Trees) (Breiman et. al., 1984).

CART, is a non-parametric technique that produces either classification or regression trees, depending on whether the dependent variable is categorical or numeric, respectively.

Trees are formed by a collection of rules based on values of certain variables in the modeling data set. Rules are selected based on how well splits based on variables' values can differentiate observations based on the dependent variable. Once a rule is selected and splits a node into two, the same logic is applied to each "child" node. Splitting stops when CART detects no further gain can be made, or some pre-set stopping rules are met.

This method appears as SimpleCART under the Trees methods in WEKA. It has been used for the current study.

In SimpleCART algorithm list of options are shown in Table 2.16.

Table 2.16 List of Options in SimpleCART

Parameter	Definition
-S	<num> Random number seed. (default 1)
-D	If set, classifier is run in debug mode and may output additional info to the console
-M	<min no> The minimal number of instances at the terminal nodes. (default 2)
-N	<num folds> The number of folds used in the minimal cost-complexity pruning.(default 5)
-U	Don't use the minimal cost-complexity pruning.(default yes).
-H	Don't use the heuristic method for binary split. (Default true).
-A	Use 1 SE rule to make pruning decision.(default no).
-C	Percentage of training data size (0-1]. (default 1).

2.5.2.3 Instance Based Classification

Instant based classifiers store a data based upon the assumption that similar samples will belong to similar groups and use it for classifying new samples of which we do not know the type. The corresponding components of an instance-based learner are the distance function which determines how similar two instances are, and the classification function which specifies how instance similarities yield a final classification for the new instance.

In this study, KStar algorithm, which is one of the sample based[10] classifiers in WEKA, has been used.

[10] Instance based, memory based and sample based are used alternatively.

KStar is an instance-based classifier, that is the class of a test instance is based upon the class of those training instances similar to it, as determined by some similarity function. It differs from other instance-based learners in that it uses an entropy-based distance function (Cleary and Trigg, 1995). Nearest neighbor algorithms (Cover and Hart, 1967) are the simplest of instance-based learners. They use some domain specific distance function to retrieve the single most similar instance from the training set.

In KStar algorithm list of options are shown in Table 2.17.

Table 2.17 List of Options in KStar

Parameter	Definition
-B	<num>Manual blend setting (default 20%)
-E	Enable entropic auto-blend setting (symbolic class only)
-M	<char> Specify the missing value treatment mode (default a)

2.5.2.4 Bootstrap Aggregating Methods

The Bagging algorithm is also a classification method classified under Meta at WEKA. Bootstrap aggregating (bagging) is a ML ensemble meta-algorithm to improve ML of classification and regression models in terms of stability and classification accuracy. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree models, it can be used with any type of model. Bagging is a special case of the model averaging approach (Breiman, 1996).

Bagging is a general technique that can be applied to numeric prediction problems as well as classification tasks. Bagging is amalgamating the various outputs into a single prediction. In bagging the models receive equal weight (Witten and Frank, 2000).

In bagging algorithm list of options are shown in Table 2.18.

Table 2.18 List of Options in Bagging

Parameter	Definition
-P	Size of each bag, as a percentage of the training set size. (default 100)
-O	Calculate the out of bag error.
-S	<num> Random number seed. (default 1)
-I	<num> Number of iterations. (default 10)
-D	If set, classifier is run in debug mode and may output additional info to the console
-W	Full name of base classifier.(default: weka.classifiers.trees.REPTree)
Options specific to classifier weka.classifiers.trees.REPTree:	
-M	<minimum number of instances> Set minimum number of instances per leaf (default 2).
-V	<minimum variance for split> Set minimum numeric class variance proportion of train variance for split (default 1e-3).
-N	<number of folds> Number of folds for reduced error pruning (default 3).
-S	<seed> Seed for random data shuffling (default 1).
-P	No pruning.
-L	Maximum tree depth (default -1, no maximum)

2.5.3 Evaluation Methods

It is necessary to have ways of predicting performance bounds in practice, based on experiments with whatever data can be obtained. There are two main ways for evaluating the researches. First of these is the separation of the data as train and test data, and to check, whether the data was separated in two groups as train and test and the correct classification of of the tested data. The second method is called Cross Validation (CV). CV[11] is a technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

In *K*-fold CV, the original sample is partitioned into *K* subsamples. Of the *K* subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $K - 1$ subsamples are used as training data. The CV process is then repeated

[11] Cross Validation : http://en.wikipedia.org/wiki/Cross-validation_(statistics)

K times (the *folds*), with each of the *K* subsamples used exactly once as the validation data. The *K* results from the folds then can be averaged (or otherwise combined) to produce a single estimation. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once. The experiments show that 10-fold CV have good results (McLachlan, et. al., 2004).

Some results are obtained after the tests done by evaluation method. After the analysis, a confusion matrix is created which shows all the results. A confusion matrix (Kohavi and Provost, 1998) contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix. Confusion Matrix is a table with the true class in rows and the predicted class in columns (Kohavi and Provost, 1998). The diagonal elements represent correctly classified compounds while the cross-diagonal elements represent misclassified compounds. The Table 2.19 also shows the accuracy of the classifier as the percentage of correctly classified compounds in a given class divided by the total number of compounds in that class. The overall (average) accuracy of the classifier is also depicted.

The entries in the confusion matrix have the following meaning in the context of our study:

- *x* is the number of **correct** predictions that an instance is **negative**,
- *y* is the number of **incorrect** predictions that an instance is **positive**,
- *z* is the number of **incorrect** of predictions that an instance **negative**, and
- *t* is the number of **correct** predictions that an instance is **positive**.

Table 2.19 Table of Confusion Matrix

		Predicted	
		Negative	Positive
Actual	Negative	**x**	**y**
	Positive	**z**	**t**

- The *Accuracy* (*AC*) is the proportion of the total number of predictions that were correct. It is determined using the equation (Kohavi and Provost, 1998):

$$AC = \frac{x+t}{x+y+z+t} \tag{2.3}$$

- The *Recall (R)* or *True Positive Rat*e (*TP*) is the proportion of positive cases that were correctly identified, as calculated using the equation (Kohavi and Provost, 1998):

$$R = \frac{t}{z+t} \tag{2.4}$$

- *Precision* (*P*) is the proportion of the predicted positive cases that were correct, as calculated using the equation (Kohavi and Provost, 1998):

$$P = \frac{t}{y+t} \tag{2.5}$$

P is the value, which indicates the accuracy ratio of the samples selected, and R is the value, which indicates the ratio of the samples selected to the samples which needs to select. The aim is to have high rates for both P and R.

In our research, the accuracy values will be examined first. Thus we will be able to see, how many of the decisions were accurate. Then we will examine the P and R values obtained based upon the algorithms listed above and examine the confusion matrix. Besides, to be able to see how effective each one of the chosen features are, all of the features will be tested with a selected algorithm and with the P and R values it will be examined which feature or features are more effective.

CHAPTER 3

TEST AND RESULTS

3.1 ACCURACY

The amount of words in terms of word sense numbers and their baseline values are listed in Table 3.1. The baseline value equals the division of the total number of sentences to word sense group with the most frequent samples. This value gives information about success rate in evaluation of the results. For example, when the word "gül" is examined, we can see two word sense groups and 200 sentences in total. 100/200: 0.5 implies the baseline value of the word "gül". For distinguishing the sentences selected for word senses one has to do a 50% distinction before every test, since there are two word sense groups. The success of the system is explained with its comparison with the standard value of %50.

Table 3.1 Number of Sentences per Words and Their Baselines

	1st sense	**2nd sense**	**3rd sense**	**Baseline (%)**
Bas	100	104	100	34
Gül	100	100	-	50
Kır	100	100	-	50
Yüz	103	100	101	50

The accuracy results of the classification processes obtained using the algorithms included in WEKA that were mentioned in the chapter 3, are listed in Table 3.2. The results are acquired according to the techniques, Train-Test (TT) and CV used for measuring the performance of the four algorithms like NB, SimpleCart, KStar and Bagging.

Table 3.2 The Accuracy Results of Algorithms (%)

	Algorithms							
	SimpleCart		**NaiveBayes**		**KStar**		**Bagging**	
Words	**T-T**	**CV**	**T-T**	**CV**	**T-T**	**CV**	**T-T**	**CV**
Bas	65.00	69.40	80.00	71.38	66.66	60.85	71.66	68.42
Gül	97.50	98.00	100.00	100.00	97.50	98.00	97.50	99.50
Kır	100.00	96.00	100.00	100.00	95.00	99.00	100.00	99.50
Yüz	86.60	86.51	90.00	85.80	85.00	81.57	85.00	86.84

3.1.1. Comparison Between Words

Looking at the results in Table 3.2 we realize that the highest accuracy values are obtained for the words “gül” and ‘kır’. For these two words, there were two different POS. In the data collected, we examined the types of these words, which were verb and noun, and the senses of the words created are word sense group for a verb and another word sense group for a noun. When the features selected for WSD listed in Table 3.12 is examined, it is seen that the type of the target word is one of these features. Therefore, since the classification algorithms classify based upon the type of the target word, we can see higher results. When the baseline values for Table 3.1 is examined, it is realized that the words “gül” and “kır” have a %50 baseline value. The NB algorithm, one of the supervised learning algorithms, rated a result of 100% for both CV methods and TT. Since the baseline value was 50%, the success here was also additive 50%.

On the other hand, SimpleCart, Bagging and NB algorithms gave an accuracy result of 100% for the word “kır”.

The word “yüz” has 3 meanings in the research. In the group marked as the first word sense, the target word is a verb, and in the second and third word senses, the word type is a noun. The best result for accuracy value for the word “yüz” was given by training test method by NB algorithm with a score of 90%. Since the baseline value of the word “yüz” is 33%, the success rate here is adding 57%.

Looking at the word “bas”, it is realized that the accuracy results for this word fell a bit more than others. For the word “bas”, data regarding 3 different word senses were collected, and all word sense groups are verbs by word type. Thus, when making classification processes for the algorithm used, the feature selected as target word type is not used at all. Here, other features have some effect. Just like the words “gül” and “kır”, since no classification regarding the word type can be done, the accuracy value for the word “bas” is fewer than the others and the baseline value for it is 34%. The highest accuracy value within the algorithms used, is the CV method of the NB algorithm with 80%. In that case, the success rate for the word “bas” is %80-%34= %46.

3.1.2. Comparison Regarding Performance Assumption Techniques

TT and CV performance assumption techniques and accuracy values for the data obtained with SimpleCart, KStar and Bagging algorithms are listed in Table 3.2. When the results are examined, it is seen that the CV method has higher values compared to TT method. In the word “bas”, SimpleCart and NB algorithms give a higher result and TT has higher results for KStar and Bagging algorithms. For the words “gül” and “kır”, CV method compared to TT method has higher values for all algorithms. But when the word “yüz” is examined, these results change. For this word, TT method in NB, SimpleCart and KStar algorithms have higher results, and in Bagging algorithm CV has a higher value. The fact that there are 3 word senses for this word group and two of them to be nouns and one of them to be a verb effected the results in this way.

In the CV method, all the data gets divided into parts as much as the value of k. The assumed k value for WEKA is 10. The data was divided into 10 parts. 9 of them were trained and one of them was reserved for test. That way, the data obtained with the CV method can be divided into 10 parts and can be tested as training and test data. In the TT method, a part of the data is reserved as training and a part of it reserved as test and these parts do not change in the following evaluation processes. However, the parts divided CV always get realized based upon the same data, so data included in the training in the previous evaluation can be in test data in the next one. This enables us to find the previously trained data to be found more easily in the test process. This is the reason why the CV method compared to TT method has higher values in our results.

Figure 3.1, Figure 3.2, Figure 3.3, Figure 3.4, Figure 3.5 and Figure 3.6 obtained from the Table 3.2. When they are examined, it is easy to see, which algorithm for which word gives better results.

When the baseline is proportional with the number of senses, the success rate rises and the results are more accurate, since there are enough sample data for different word senses.

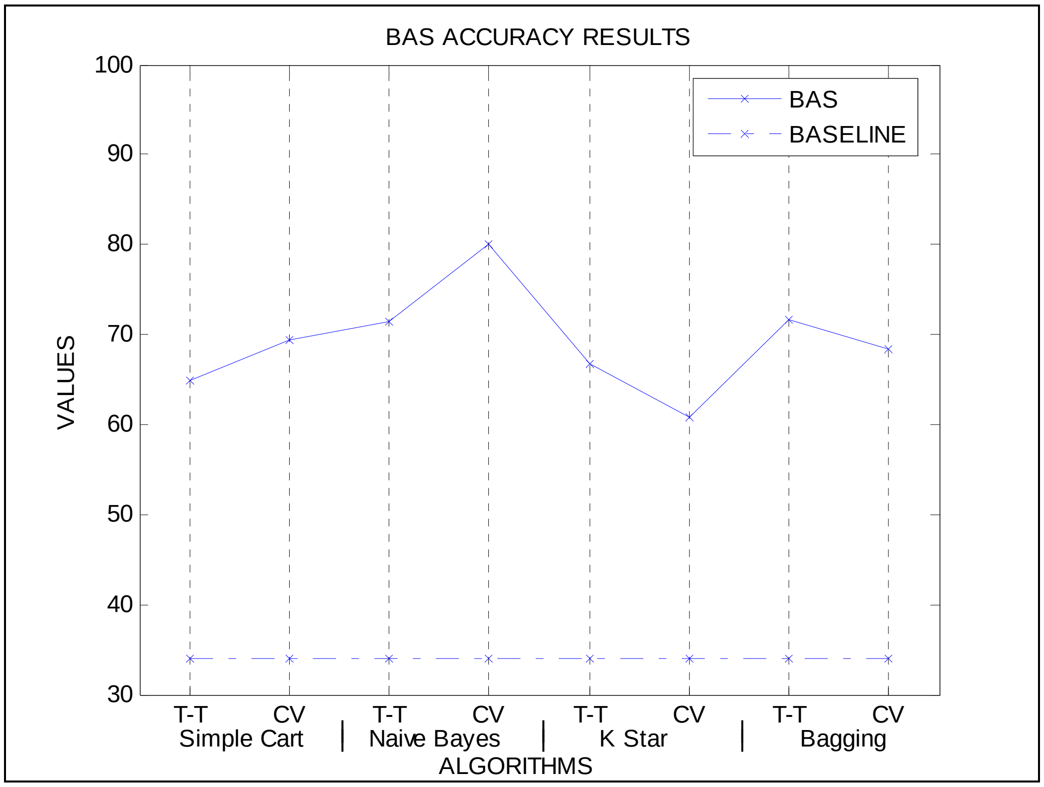


Figure 3.1 The Accuracy Results of Word "Bas"

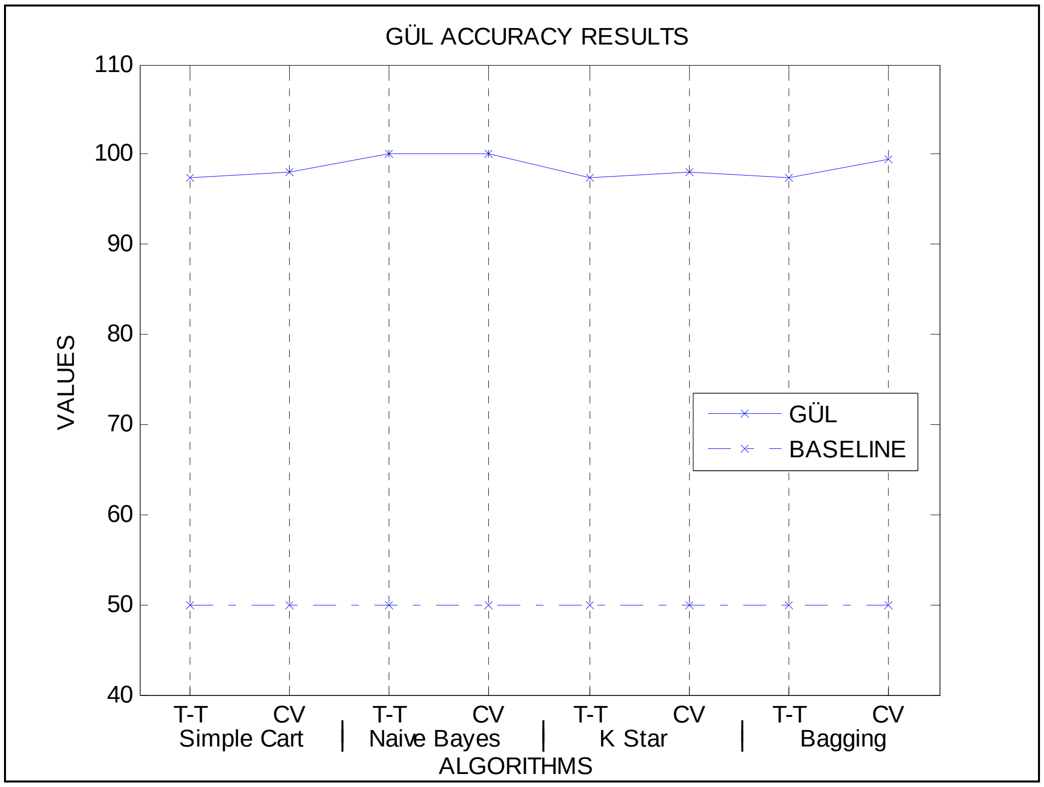


Figure 3.2 The Accuracy Results of Word "Gül"

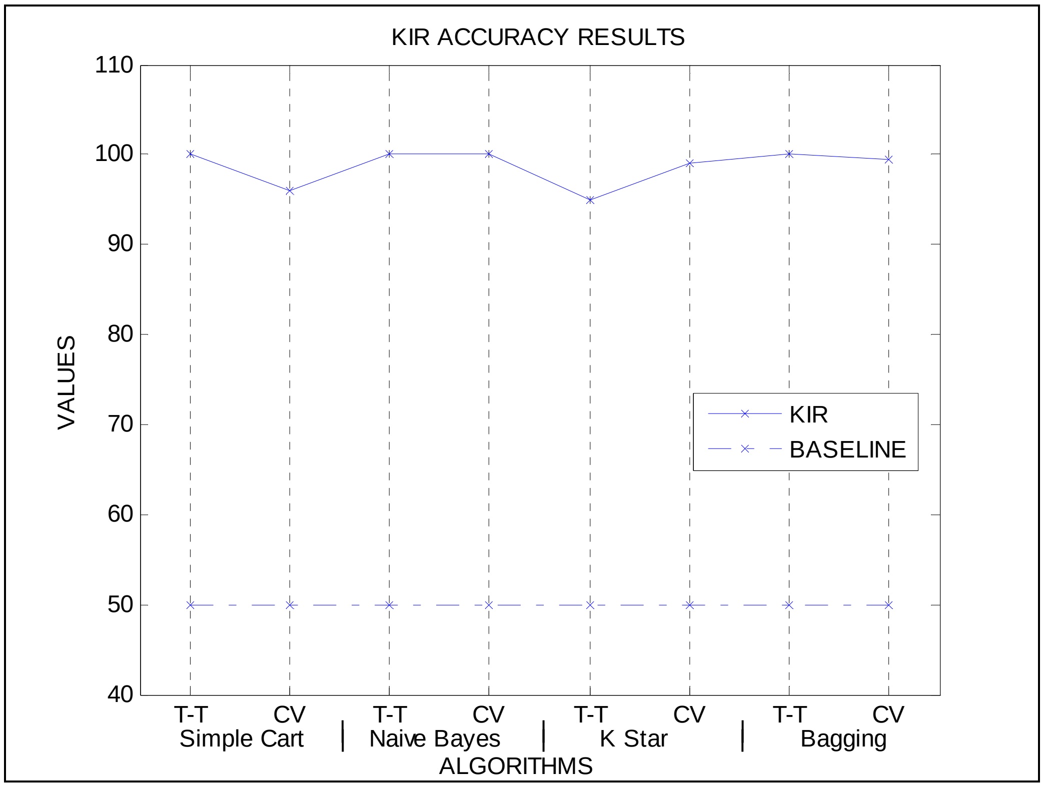


Figure 3.3 The Accuracy Results of Word "Kır"

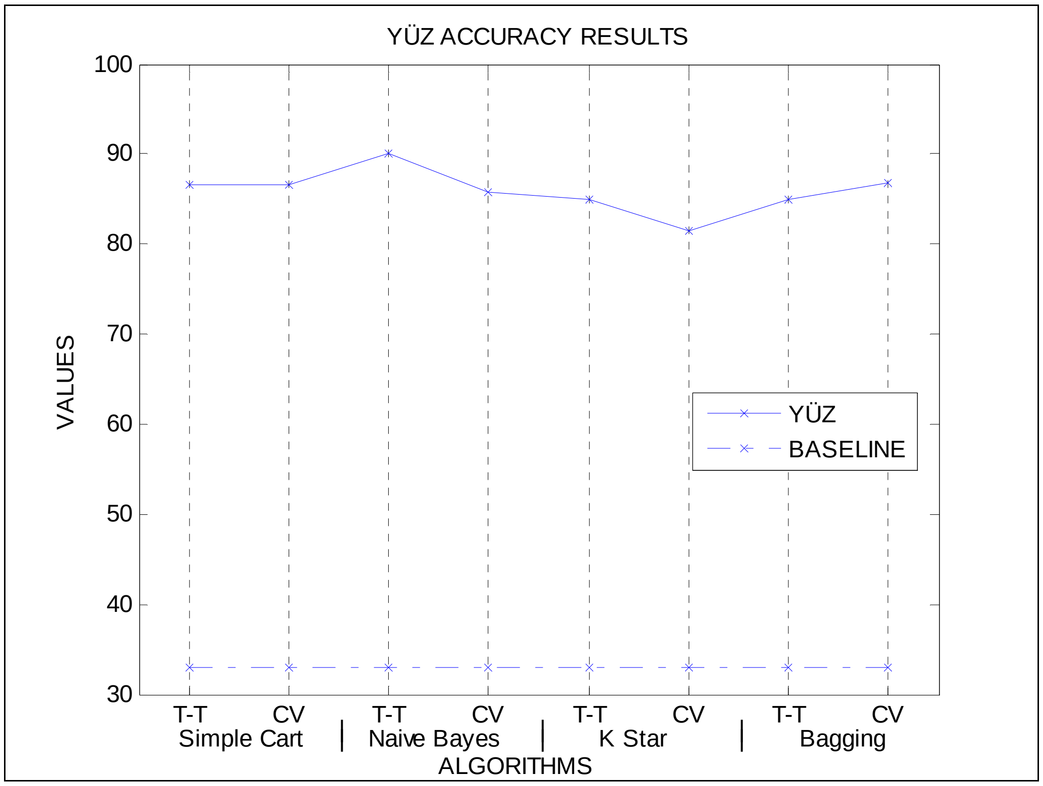


Figure 3.4 The Accuracy Results of Word "Yüz"

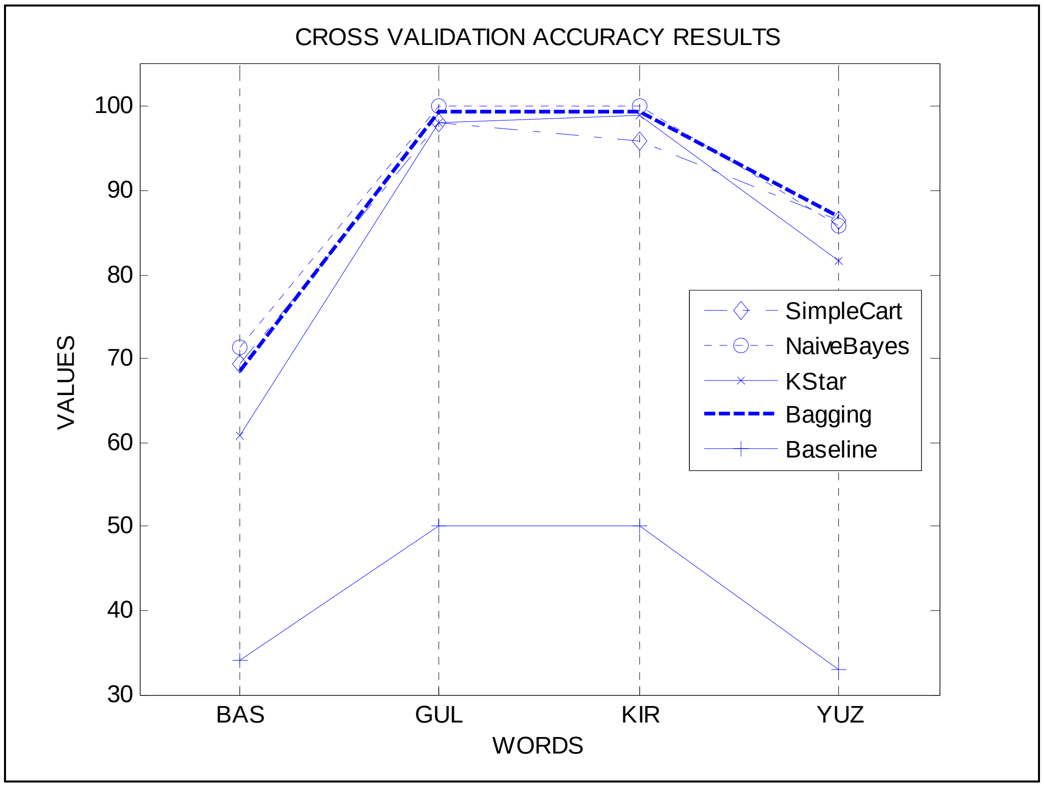


Figure 3.5 Cross Validation Accuracy Results

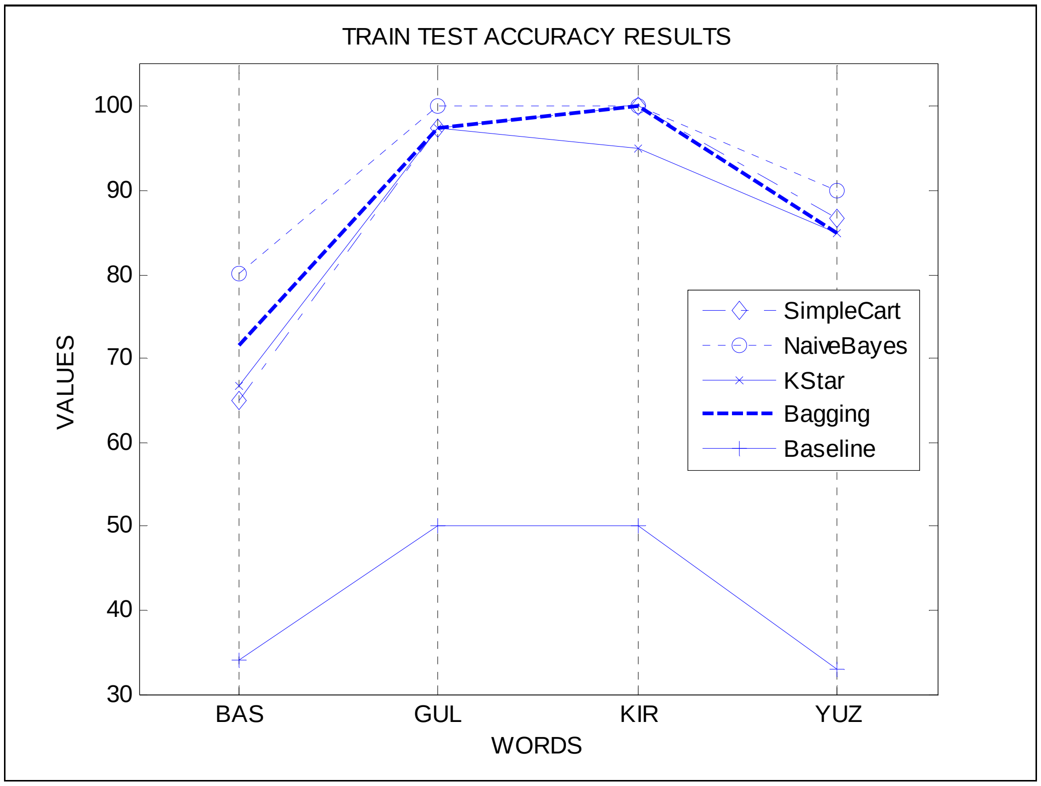


Figure 3.6 Train-Test Accuracy Results

3.2 CONFUSION MATRIX AND PRECISION-RECALL RESULTS

The Confusion Matrix (CM) in the classification processes made in WEKA based upon the selected algorithms in the target words are listed in Table 3.3,Table 3.5, Table 3.6 and Table 3.7. Using this CM, P and R values can be obtained. The Table 3.8, Table 3.9, Table 3.10 and Table 3.11 show the P and R values obtained from the CM. In the tables, a, b, c represent the word senses. The statement a=1 shown in the lines imply, that the symbolic value was used for the word sense 1. The lines show the sum of the data regarding that class. The columns show the values found by the methods.

Table 3.3 Confusion Matrix of the Word "Bas"

Algorithms	**T-T**				**CV**			
SimpleCart	a	b	c		a	b	c	
	10	0	10	a=1	67	2	31	a=1
	2	14	4	b=2	9	73	22	b=2
	1	4	15	c=3	15	14	71	c=3
NaiveBayes	a	b	c		a	b	c	
	13	1	6	a=1	76	4	20	a=1
	0	20	0	b=2	10	79	15	b=2
	1	4	15	c=3	23	15	62	c=3
KStar	a	b	c		a	b	c	
	13	1	6	a=1	67	7	26	a=1
	3	17	0	b=2	15	72	17	b=2
	3	7	10	c=3	31	23	46	c=3
Bagging	a	b	c		a	b	c	
	9	4	7	a=1	70	5	25	a=1
	2	18	0	b=2	9	78	17	b=2
	0	4	16	c=3	22	18	60	c=3

It is better to show how the Accuracy, P and R values are calculated with the obtained CM using the values in Table 3.3.

The matrix obtained with NB algorithm and CV method for the word “bas” is shown in Table 3.4.

Table 3.4 The Confusion Matrix of the Word “Bas” for NaiveBayes – CV

a	b	c		a	b	c	
x	y	w	a=1 b=2 c=3	76	4	20	a=1 b=2 c=3
z	t	s		10	79	15	
k	l	m		23	15	62	

(a) (b)

The accuracy formula was given in (2.3). Since there are 3 classes for the word “bas”, the CM for this word will be like in Table 3.4(b). According to AC formula the result is:

$$AC = \frac{x+t+m}{x+y+w+z+t+s+k+l+m} = \frac{76+79+62}{76+4+20+10+79+15+23+15+62} = 71.38$$

When the CM of the word “bas” is examined, it is seen that the second word sense was classified the most correctly. For example, in the SimpleCart method, 73 of 104 sentences were marked as word sense number 2 and 31 of them were not to be put in the right class. Again, using the NB method, 79 of 104 sample sentences were classified in word sense number 2, and 25 of them were not be able to be classified correctly. Again in the CM, the word sense number 1 was found close word sense number 3, and was classified in the word sense number 3, and similarly, in the sentences with the word sense number 3, word sense number 1 than to word sense number 2. The second meaning implies using force towards something and pushing it, putting the foot on something, and thus it was easier to distinct it from the word sense number 3, which implies to spread and the word sense number 1 which implies pulling something in one direction by force and covering the environment.

When the CM of the word "gül" in Table 3.5 is examined, it is seen that the word sense number 2 is more correctly classified than word sense number 1. The suffixes word sense number 2, which is a noun, get have an important effect on classifying the word.

Table 3.5 Confusion Matrix of the Word "Gül"

Algorithms	**T-T**			**CV**		
SimpleCart	a	b		a	b	
	19	1	a=1	96	4	a=1
	0	20	b=2	0	100	b=2
NaiveBayes	a	b		a	b	
	20	0	a=1	100	0	a=1
	0	20	b=2	0	100	b=2
KStar	a	b		a	b	
	19	1	a=1	97	3	a=1
	0	20	b=2	1	99	b=2
Bagging	a	b		a	b	
	19	1	a=1	99	1	a=1
	0	20	b=2	0	100	b=2

When the CM of the word "kır" in Table 3.10 is examined, it is seen that it is highly classified correctly due to its type and suffixes and due to the selection of two words. Only in SimpleCart algorithm, 7 of the word sense number 1 was marked as the word sense number 2.

When we examine the CM of the word "yüz" in Table 3.7, we see that the word sense number 3 was classified more correctly, because word sense number 1 and 2 are noun and the third one is a verb. This result indicates how important the word type for WSD is. Word sense number 1, in comparison with word sense number 3, was classified more like the word sense number 2, and similarly the word sense number 2 was classified more like the word sense number 1 in comparison with the word sense number 3. But the classification of the word sense number 1 is more correct than the word sense number 2. Here it is seen that the word sense number 1 is used as a number and thus was easier to classify in compared to other noun word senses.

Table 3.6 Confusion Matrix of the Word “Kır”

Algorithms	**T-T**			**CV**		
SimpleCart	a	b		a	b	
	20	0	a=1	93	7	a=1
	0	20	b=2	1	99	b=2
NaiveBayes	a	b		a	b	
	20	0	a=1	100	0	a=1
	0	20	b=2	0	100	b=2
KStar	a	b		a	b	
	19	1	a=1	99	1	a=1
	1	19	b=2	1	99	b=2
Bagging	a	b		a	b	
	19	1	a=1	99	1	a=1
	0	20	b=2	1	99	b=2

Other results that can be obtained using the CM are the P and R values. Now it can be seen how the variables P and R in CV method and NB algorithm and P R values of the word “bas” can be calculated in Table 3.4.

Table 3.7 Confusion Matrix of the Word “Yüz”

Algorithms	**T-T**				**CV**			
SimpleCart	a	b	c		a	b	c	
	20	0	0	a=1	102	0	1	a=1
	8	12	0	b=2	4	96	0	b=2
	0	0	20	c=3	3	0	98	c=3
NaiveBayes	a	b	c		a	b	c	
	20	0	0	a=1	88	14	1	a=1
	5	14	1	b=2	26	73	1	b=2
	0	0	20	c=3	1	0	100	c=3
KStar	a	b	c		a	b	c	
	20	0	0	a=1	90	12	1	a=1
	7	11	2	b=2	37	60	3	b=2
	0	0	20	c=3	1	2	98	c=3
Bagging	a	b	c		a	b	c	
	20	0	0	a=1	91	12	0	a=1
	8	11	1	b=2	23	74	3	b=2
	0	0	20	c=3	2	0	99	c=3

P and R formula were given in (2.4) and (2.5). When the data in Table 3.4 is used, the P values of the word sense numbers 1, 2 and 3 can be calculated with the formula given below:

$$P_1 = \frac{x}{x+z+k} \;,\; P_2 = \frac{t}{y+t+k} \;,\; P_3 = \frac{m}{m+s+w}$$

The P values for the meanings of the word “bas” can be calculated like this:

$$P_1 = \frac{76}{76+10+23} \cong 0{,}7$$

$$P_2 = \frac{79}{4+79+15} \cong 0{,}8$$

$$P_3 = \frac{62}{20+15+62} \cong 0{,}63$$

The R values for the word senses of the word “bas” can be calculated like this:

$$R_1 = \frac{x}{x+y+w} = 0{,}76$$

$$R_2 = \frac{t}{z+t+s} \cong 0{,}76$$

$$R_3 = \frac{m}{k+l+m} = 0{,}62$$

All results obtained like this are listed in Table 3.8.

Table 3.8 Precision Recall Values for the Word “Bas”

	Algorithms	**SimpleCart**		**NaiveBayes**		**KStar**		**Bagging**	
		T-T	**CV**	**T-T**	**CV**	**T-T**	**CV**	**T-T**	**CV**
P	**1**	0.77	0.74	0.93	0.70	0.69	0.60	0.81	0.70
	2	0.78	0.82	0.80	0.80	0.68	0.70	0.70	0.78
	3	0.52	0.58	0.71	0.63	0.63	0.52	0.70	0.59
R	**1**	0.50	0.67	0.65	0.76	0.65	0.67	0.45	0.70
	2	0.70	0.70	1.00	0.76	0.85	0.69	0.90	0.75
	3	0.75	0.71	0.75	0.62	0.50	0.46	0.80	0.60

When the P and R results for the word “bas” are examined, it is noted that the P value for the word sense number 1 with NB algorithm and TT method is 0,93. For the word sense number 2, we see that the P value for SimpleCart algorithm and CV method is 0.82 and for the word sense number 3 we see that the result is 0.71 with NB algorithm and TT method. When the low P values are examined, a value of 0.50 is noticed. The P value to be low indicates that in the sentences that are considered to be included in their correct class a few sentences are in their correct class. Looking at the R results, it is seen that for the first word, using the NB and CV method a result of 0.76 is obtained and using the NB algorithm and TT method a result of 1 is obtained. For the word sense number 3, the Bagging algorithm with the TT method found a result of 0.8. These results indicate the ratio of selected samples to samples that needs to be selected. This means, all of the samples selected with the TT method for the word sense number 2 are classified correctly. Furthermore, the lowest R value is the Bagging method and TT method of the word sense number 1with a result of 0.45. This indicates that the numbers of sentences that are not classified as in the classes where they should have been are high.

Table 3.9 Precision Recall Values for the Word “Gül”.

	Algorithms	**SimpleCart**		**NaiveBayes**		**KStar**		**Bagging**	
		T-T	**CV**	**T-T**	**CV**	**T-T**	**CV**	**T-T**	**CV**
P	**1**	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00
	2	0.95	0.92	1.00	1.00	0.95	0.97	095	0.99
R	**1**	0.95	0.962	1.00	1.00	0.95	0.97	0.95	0.99
	2	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00

When the P – R values in Table 3.9 are examined, which is obtained from the CM for the word “Gül”, a classification of 1.00 P value was made with NB algorithms in both methods. Besides, the P values of word sense number 1 and R values of word sense number 2 are 1.00. The P score of 1.00 means that every result retrieved by a search was relevant and the R score of 1.00 means that all relevant documents were retrieved by the search.

Table 3.10 Precision Recall Values for the Word “Kır”

	Algorithms	**SimpleCart**		**NaiveBayes**		**KStar**		**Bagging**	
		T-T	**CV**	**T-T**	**CV**	**T-T**	**CV**	**T-T**	**CV**
P	**1**	1.00	0.99	1.00	1.00	0.95	0.99	1.00	0.99
	2	1.00	0.93	1.00	1.00	0.95	0.99	1.00	1.00
R	**1**	1.00	0.93	1.00	1.00	0.95	0.99	1.00	1.00
	2	1.00	0.99	1.00	1.00	0.95	0.99	1.00	0.99

In the word “Kır”, all sentences were classified appropriately with the NB algorithm. Besides, it is seen that the SimpleCart and Bagging algorithms have P and R values of 1.00 with the T-T method.

Table 3.11 Precision Recall Values for the Word "Yüz"

	Algorithms	SimpleCart		NaiveBayes		KStar		Bagging	
		T-T	CV	T-T	CV	T-T	CV	T-T	CV
P	1	0.71	0.79	0.80	0.76	0.74	0.70	0.71	0.79
	2	1.00	0.82	1.00	0.84	1.00	0.81	1.00	0.86
	3	1.00	1.00	0.95	0.98	0.91	0.96	0.95	0.97
R	1	1.00	0.83	1.00	0.85	1.00	0.87	1.00	0.88
	2	0.60	0.76	0.70	0.73	0.55	0.60	0.55	0.74
	3	1.00	1.00	1.00	0.99	1.00	0.97	1.00	0.98

For the word sense number 1 of the word "yüz" 0.8 P result was obtained with NB algorithm and T-T method. For the word sense number 2, value of 1.00 was obtained with each 4 algorithms' T-T method. For the word sense number 3, value of 1.00 was obtained with SimpleCart algorithm. Looking at the R values, it is seen that all four methods cover the selected examples that need to be selected for word sense numbers 1 and 3. However, for the word sense number 2, value of 0.76 was obtained with the SimpleCart algorithm and CV method. Since the word sense numbers 1 and 2 were in noun form and the word sense number 3 was in verb form, R values of the word sense number 3 are higher than the other senses. In the CM, especially the word sense number 3 class was classified easily, and higher values were obtained in the word sense number 1 compared to word sense number 2. The first sense of "yüz" is the number. The number sense of the word "yüz" is classified more easily than the noun sense of the word.

3.3 TESTS REGARDING FEATURE SELECTION

In Chapter 2, the information about which features could be effective in word sense selection is given. In Table 3.12 below, the features selected for the study are given with their abbreviations.

Table 3.12 Selected Features

ABBREVIATIONS	FEATURES
TPP	Part Of Speech of Two Preceding Word
TPS	Suffix of Two Preceding Word
PP	Part Of Speech of Preceding Word
PS	Suffix of Preceding Word
TP	Part Of Speech of Target Word
TS	Suffix of Target Word
SP	Part Of Speech of Succeeding Word
SS	Suffix of Succeeding Word
TSP	Part Of Speech of Two Succeeding Word
TSS	Suffix of Two Succeeding Word
S	Sense

We determined that we obtained the best results with NB algorithm with the accuracy, P and R values obtained as a result of the algorithms we selected for our target words. In this part of the study, we observed how effective the features were on the words that we wanted to disambiguate. For each of our target words, each feature was selected separately and in the table below, the effectiveness of each feature with NB Algorithm and CV method is shown in Table 3.14 and Table 3.15, P and R values are calculated. In the tables, the highest, lowest and average P and R values obtained for the target words with 10 features, except for the sense class, are shown. To ensure easier comprehension of the features' effect on the table, if the obtained value is closer to the maximum value, it was marked with double underbars; if the obtained value is closer to minimum, it was marked with crossbar, and if it was close to the average value, it was marked as is. It is shown in Table 3.13.

When P values in Table 3.14 are examined, it is seen that the words, which have two sense groups, are of very high value. Besides, for each of the 4 target words, the suffix of the target word has a high value in WSD. These values are 0.985 for the word "gül", 0.98 for the word "kır", 0.836 for the word "yüz" and 0.602 for the word "bas". This shows that the suffix of the target has an important effect on WSD. While the suffix of the preceding word “bas” had an important effect on WSD, in the other, it has an average importance. The type and suffix of the word succeeding the target word has an average importance on WSD.

When drawn away from the target word, it is observed that the selected feature’s effect decreases. For instance, the type and suffix of the two preceding word is not very effective on the words “kır” and “yüz”. The two succeeding words type is not effective on the word “gül”, however, it has an average value for the other 3 words. The value of the suffix of the two preceding word is close to minimum for the words “gül”, “kır”, and “yüz”.

As mentioned before, since the words “gül” and “kır have 2 senses and due to the effect of the word type in WSD, the other feature’s effect decrease, especially when drawn away from the target word. However, since all 3 senses of the word “bas” are in verb type, the types do not have any effect at all, so, the other features’ effects on WSD increase. 6 out of 10 features are close to maximum value for the word “bas”, and 2 of them are close to maximum for the words “gül” and “kır” and 2 for the word “yüz”.

When the word “yüz” is examined two senses for selected for noun type, and one for verb type. The suffix of the word has the highest value in WSD. This is followed by the suffix of the succeeding word, type of the two succeeding word, suffix and type of the preceding word.

In the light of these results, it can be said that, the high number of these features don’t have a direct proportion of value in WSD, and determining in reducing the time spent for calculation the effective features in WSD.

Table 3.13 Type of Representation

Type of Representation	**Example**	**Convergence**
Double underbar	$\underline{\underline{0.98}}$	Close to maximum value
Crossbar	~~0.55~~	Close to minimum value
Unmarked	0.67	Close to average value

Table 3.14 Precision Results

	BAS	GÜL	KIR	YÜZ
TPP	0.43	0.67	~~0.64~~	~~0.49~~
TPS	0.41	0.67	~~0.62~~	~~0.45~~
PP	0.39	0.63	0.73	~~0.46~~
PS	0.58	0.66	0.68	0.59
TP	~~0.12~~	0.99	0.99	0.50
TS	0.60	0.98	0.98	0.84
FP	0.37	0.69	0.85	~~0.48~~
FS	0.44	0.65	0.77	0.60
TFP	0.34	~~0.55~~	0.73	0.61
TFS	0.27	~~0.53~~	~~0.61~~	~~0.44~~
MAX	**0.60**	**0.99**	**0.99**	**0.84**
MIN	**0.12**	**0.53**	**0.61**	**0.44**
AVR	**0.34**	**0.70**	**0.76**	**0.55**

Table 3.15 Recall Results

	BAS	GÜL	KIR	YÜZ
TPP	0.42	0.65	~~0.63~~	0.51
TPS	0.40	0.64	~~0.60~~	~~0.41~~
PP	0.36	0.62	0.72	0.48
PS	0.58	0.64	0.67	0.53
TP	~~0.34~~	0.99	0.99	0.67
TS	0.60	0.98	0.98	0.73
FP	0.37	0.65	0.80	0.53
FS	0.40	0.63	0.75	0.55
TFP	~~0.34~~	~~0.55~~	0.71	0.54
TFS	~~0.30~~	~~0.52~~	~~0.58~~	~~0.43~~
MAX	**0.60**	**0.99**	**0.99**	**0.73**
MIN	**0.30**	**0.52**	**0.58**	**0.41**
AVR	**0.41**	**0.69**	**0.74**	**0.54**

CHAPTER 4

DISCUSSION AND CONCLUSION

In this section, the problems encountered during WSD study and results of the study are discussed and future studies have been mentioned.

4.1 PROBLEMS ENCOUNTERED

Turkish is a very rich language. Even the names of the relatives, which are seen under one single word in the other languages, are different in Turkish. Besides, there is a large number of color names in Turkish. Due to the agglutinative nature of Turkish and the variety of suffixes, senses of the words change and their number of senses increase. For this reason, it is quite difficult to carry out a study on disambiguation of words in such a rich language.

Several problems have been encountered during our study on WSD for Turkish words. Problems arise in the phase of selecting data to study at the beginning. These problems are as follows:

- While searching for the senses of a specific word, there are differences between the dictionaries. The number of senses may not be the same.
- One of the factors that should be considered is, whether all the words that belong to a language or only a particular group of words will be sense disambiguated. Because, studying all words may require a long process and work load. Selecting sample words and determining particular features on these words and then making generalizations with this data may not bring accurate results.

- Another factor that changes the results is selecting words that can be distinguished as difficult or easy.

Some problems arising from the language structure were encountered.

- Some *words used in wrong senses* were encountered in sentences.
- One of the factors that affect WSD studies is the word being used in figurative or real sense.

Data processing phase is an important process that affects the studies as well. The problems encountered in this process are as follows:

- First of all, the features that will affect the word sense should be selected for the collected data. However, it is difficult to decide what these features will be. Making generalizations regarding features for all words affect values of the results.
- Selecting the methods to be implemented after feature selection is an important phase as well. Different algorithms are used in WSD studies. Among these algorithms, it is not possible to come up with a result showing the best algorithm that distinguishes the senses of all words in the highest value.

One of the most important problems that was encountered during our studies is the inadequacy of database or processed texts that can be used by those who would like to carry out WSD studies for Turkish words. It was quite a long process to establish the database composed of sentences that involve the target words. Allowing the data that was collected in previous studies available to common use will both help eliminate the data collection process, and also allow the results, which were obtained from a common database, to be evaluated in comparison with each other.

4.2 RESULTS AND FUTURE STUDIES

First of all, along with these problems, we decided on the selection of words that we would like to disambiguate in our study. During the selection of words, rich verb structure of Turkish was taken into consideration and the purpose was selecting words that have a verb sense. Besides verbs, nouns were included in the study too. Thus, the effectiveness of the word type as a distinctive feature was researched. In this direction, the words that were chosen, the word "basmak" which has 3 different senses, the words "gül", "kır" which can be used both as a verb and a noun and the word "yüz" which has 3 senses.

After selecting the words, TDK dictionary was scanned and the words' senses were examined for sense disambiguation. With the coarse grained application, which was done within WSD studies and also used in SENSEVAL projects, some senses were combined and a group of sense was established. Thus, the words' most frequently used senses were selected.

Sentences for the target words in accordance with senses were selected. The sentences in the study were selected from different sources by scanning one by one. Instead of taking the all sentences in which the words we are looking for in a database in this study were used; selecting the sentences, which meet the target word's sense provided more clear results for distinguishing the senses from each other. In the following phase, we obtained results by making necessary root and suffix disjunctions with Zemberek program, providing resolution of words with Zemberek Parser program, which hadn't been accurately analyzed before, and carried out tests with the selected algorithms. In accordance with these results, it was found that the type of the word had an important effect on the WSD. For instance, two sense groups were established for each words "gül" and "kır". In one of these senses, the word was used as a noun and in the other, it was used as a verb. In different algorithms for the words *Gül and kır,* Accuracy values of %95 and above were obtained. For the word type "*Bas",* Accuracy values between %65 and %80 were obtained from the sentences in which only verb type were selected. For the word *Yüz,* two senses for noun form and one sense for verb form were selected. The values obtained from this study vary between %81 and %90.

Supervised learning algorithms were used in the study. The data were evaluated separately as test and train so as to measure the effects of different evaluation methods, and also evaluations were made with CV method. The best results are obtained with NaiveBayes algorithms in these two evaluations.

When we measured how effective the selected features were in WSD, we observed that the most effective feature was the type of the word, and this was followed by the suffix on the target word, the preceding and succeeding words' types and their suffixes. When we look at the selected features, we can see that these features' effect on WSD decreases as we draw away from the target word. As with the word *"bas",* if the word type is not a distinctive feature, in this case, it was observed that types of the two preceding and the succeeding word of the target word and their suffixes had effect on the WSD.

Each sense of the word was considered as separate for the data collected in our study. If a sense of the target word, which hasn't been used before, was to be included in the study; instead of repeating the whole study for this application, only the part to be added was processed, and this is a progressive method for the study.

We didn't examine structures such as idiomatic phrases involving the target words in feature selection. In the following studies, determining these word phrases and finding the words that are frequently used with the target words may be effective features for WSD. Or, while searching for the sense of the word, there might be words that are frequently used with these words in the sentences. These may be a method to be used in WSD too. For instance, if the words such as *deniz, havuz, su* are used in sentences in which the word *yüz* is used, in this case, it might be understood that the verb *yüzmek* is meant.

One of the points to be considered in future studies should be determining the words that are frequently used in the sentences in which the target word is used. If these words are used in the sentences in which every sense of the word is used, in this case, they may be excluded from the study since they don't have any effect on the sense.

The sentence analyzing program called Zemberek is source code based and it provides target programs to be developed via these codes, so, this is supporting for these studies.

REFERENCES

Agirre, E., Ansa, O., Martinez, D., Hovy, E., “Enriching Wordnet Concepts with Topic Signatures”, *Proceedings of the NAACL Workshop on Wordnet and Other Lexical Resources: Applications, Extensions and Customizations*, Pittsburg, USA, 3 -4 June 2001 pp.123-132., 2001.

Agirre E., Edmonds P. G., “Word Sense Disambiguation Algorithms and Application”, *SpringerVerlag*, Western Europe, 2006.

Aytekin, C., Say A. C. C., Akçok E., "ELIZA speaks Turkish: A Conversation Program for an Agglutinative Language," *Üçüncü Türk Yapay Zeka ve Yapay Sinir Aglari Sempozyumu*, pp. 435. Ankara, 1994.

Basili R., Rocca M. D., Pazienza M. T., “Contextual Word Sense Tuning and Disambiguation”, *Applied Artificial Intelligence*, Vol.11, No. 3, pp. 235-262, 1997.

Blau H., McGovern A., “Categorizing Unsupervised Relational Learning Algorithms”, *The Workshop on Learning Statistical Models from Relational Data at International Joint Conference on Artificial Intelligence,* August 9-15, Acapulco- Mexico, 2003.

Bozşahin, C., “The Combinatory Morphemic Lexicon”, Computational *Linguistics*, Vol.28, No:2, pp. 145-186, 2002.

Breiman, L., “Bagging Predictors”, *Machine Learning*, Vol.24, No.2, pp. 123-140, 1996.

Breiman, L., Friedman, L., Olshen, R., Stone, C., *Classification And Regression Trees*, Wadsworth Inc., Belmont, California, 1984.

Brown, P.F., Della Pietra, S., Della Pietra, V., Mercer, R.L., “Word Sense Disambiguation Using Statistical Methods”. *In Proceedings Of The 29th Annual Meeting Of The Association For Computational Linguistics (ACL),* pp.264-270, 1991.

Cleary, J. G., Trigg, L. E., "K*: An Instance-Based Learner Using an Entropic Distance Measure", *Proceedings Of The 12th International Conference On Machine Learning*, Tahoe City, California, USA, July 9-12, 1995, pp.108-114, 1995.

Cover, T.M., Hart, P.E., "Nearest Neighbor Pattern Classifcation", IEEE *Transactions on Information Theory*, Vol.13, pp. 21-27. 1967.

Demir, Ş. *Improved Treatment of Word Meaning in a Turkish Conversational Agent*, M.S. Thesis, Boğaziçi University. 2003.

Duda, R. O. and Hart P. E., *Pattern Classification and Scene Analysis*, John Wiley and Sons New York, 1973.

Edmons, P., "SENSEVAL: The Evaluation of Word Sense Disambiguation Systems", *ELRA Newsletter*, Vol. 7 No. 3, pp.5-14, 2002.

Hankamer, J., "Finite State Morphology and Left-To-Right Morphology", *West Coast Conference on Formal Linguistics*. Stanford, 1986.

Haruno, M., Shirai, S., Ooyama, Y., "Using Decision Trees To Contruct A Practical Parser", *Proceedings of the Joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL),* Montreal-Canada, August 10-14, 1998, pp. 1136-1142, 1998.

Hirst, G., *Semantic Interpretation and the Disambiguation of Ambiguity*, Cambridge University Press, England, 1987.

Ide, N., Veronis, J., "Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art", *Computational Linguistics*, Vol. 24, No.1, pp. 1-40, 1998.

Jurafsky, D., Martin, J.H., *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition,* Prentice Hall, USA, 2000.

Kardeş, O., *Bir Uygulama Alanında Türkçe Metnin Anlambilimsel Gösterimi,* M.S. Thesis, Boğaziçi University, 2002.

Kilgarriff, A., Rosenzweig, J., 2000, "Introduction to the Special Issue on Evaluating Word Sense Disambiguation Systems", *ACM* Vol. 8, No. 4, pp. 279 – 291, 2002.

Kohavi R., Provost F., "Special Issue on Applications of Machine Learning and the Knowledge Discovery Process", *Machine Learning*, Vol.30, No.2-3, pp. 127-132 , 1998.

Köksal, A., *Türkçe'nin Özdevimli Biçimbirim Çözümlemesi*, Ph.D. Thesis, Hacettepe University, 1976.

Marquez, L., *Part-Of-Speech Tagging: A Machine Learning Approach Based on Decision Trees*, Ph.D. Thesis, Universitat Politecnica De Catalunya, 1999.

McLachlan, G. J., Do, K.A., Ambroise, C., *Analyzing microarray gene expression data*. Wiley-Interscience, New Jersey, U.S.A., 2004.

Mihalcea, R., "Instance Based Learning With Automatic Feature Selection Applied to Word Sense Disambiguation", *Proceedings of the 19th International Conference On Computational Linguistics (COLING 2002)*, Taiwan, August 2002, pp. 202-214, 2002.

Money, R. J., "Encouraging Experimental Results on Learning CNF", *Machine Learning*, Vol.19 No.1 pp.79-92, 1995.

Oflazer, K., "Two-Level Specification of Turkish Morphology"*, Literary and Linguistic Computing,* Vol. 9, No.2, pp. 137-148, 1994.

Orhan Z., Altan Z., "Makine Öğrenme Algoritmalarıyla Türkçe Sözcük Anlamı Açıklaştırma"*, MakinaTek,* Vol. 92, pp. 100-104, 2005.

Orhan Z., Türkçe *Metinlerdeki Anlam Belirsizliği Olan Sözcüklerin Bilgisayar Algoritmalari ile Anlam Belirginleştirmesi,* Ph.D. Thesis, İstanbul University, 2006.

Özgür, A., *Belge Sınıflandırma için Denetimli ve Denetimsiz Öğrenme Algoritmaları*, M.S. Thesis, Boğaziçi University, 2004.

Pembe, F.C., *A Linguistically Motivated Information Retrieval System for Turkish*, M.S. Thesis, Boğaziçi University, 2004.

Quinlan J. R. “Induction of Decision Trees”*, Machine Learning*, Vol.1, No.1, pp. 81-106, 1986.

Rumelhart, D. E, Hinton, G.E., Williams, R. J., “Learning Invernal Representations by Error Propagation”, In Rumelhart, D.E. and McClelland, J. L. (Eds), *Paralel Distributed Processing*, Vol. 2 pp. 318-362, 1986.

Schrater, P., Sundareswara, R., “Theory and Dynamics of Perceptual Bistability”, *Advances in Neural Information Processing Systems,* Vol. 19, pp. 1217-1224, 2006.

Stoop, A., “TRANSIT in the World of Machine Translation: Towards an Automatic Translator for Dutch and Turkish”, *Proceedings of the Third Conference on Turkish Linguistics*, pp. 78-85, Tilburg, Hollanda 1987.

Tanaka, H., “Decision Tree Learning Algorithm with Structural Attributes: Application To Eylemal Case Frame Acquisition”, *Proceedings of the 16th International Conference on Computational Linguistics (COLING),* Copenhagen, Denmark, August 5-9, 1996, pp. 943-948, 1996.

Weaver, W., *Translation*, Mimeographed, pp. 12, July 15, 1949. Reprinted In Locke, William N. Ve Booth, A. Donald, 1955 (Eds.), Machine Translation of Languages, John Wiley & Sons, New York, pp.15-23. 1949.

Weiss, S.M., Apte, C., Damerau, F.J., Johnson, D.E., Oles, F.J., Goetz, T., Hampp, T., “Maximizing Text-Mining Performance”, *IEEE Intelligent Systems*, Vol.14, No.4, pp. 63-69, 1999.

Weizenbaum, J., “ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine”, *Association for Computing Machinery (ACM)*, Vol. 9, No. 1, pp. 36-45, 1966.

Witten, I. H., Frank, E., *Data Mining Practical Machine Learning Tools and Techniques with Java Implementations,* Academic Press, USA, 2000.

APPENDIX A

Main Program

```
/** This code was written to prepare weka input data file for word sense disambiution(WSD) project.
 * this class for GUI.**/
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;

public class anaProgram extends JPanel{

	JFrame frm;
	JPanel pane;
	JFileChooser chooser, chooser2;
	JButton btn_selectedFile, btn_selectedFolder, btn_clearFile, btn_clearFolder;
	JButton btn_parse, btn_corpus;
	JTextField tf_word, tf_selectedFile, tf_selectedFolder;
	JLabel lbl_word, lbl_selectedFile, lbl_selectedFolder, lbl_operation;
	anaProgram(){
		makeView(); makeFrame();
	}
	public class Actions  implements ActionListener {

		public void actionPerformed(ActionEvent e) {
			if(e.getSource() == btn_selectedFile)
			{
				int returnVal = chooser.showOpenDialog(pane);
			    if(returnVal == JFileChooser.APPROVE_OPTION) {
				tf_selectedFile.setText(chooser.getSelectedFile().getAbsolutePath());
			    }
			}
			else if(e.getSource() == btn_clearFile)
			{
				tf_selectedFile.setText("");
			}
			else if(e.getSource() == btn_selectedFolder)
			{
				chooser2.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
				int returnVal = chooser2.showOpenDialog(pane);
			    if(returnVal == JFileChooser.APPROVE_OPTION) {
				tf_selectedFolder.setText(chooser2.getSelectedFile().getAbsolutePath());
			    }
			}
			else if(e.getSource() == btn_clearFolder)
			{
					tf_selectedFolder.setText("");
				}
				else if(e.getSource() == btn_parse)
				{
					String word = getWord();
					if(word != null)
						word = word.trim();
					String file = getSelectedFile();
					String folder = getSelectedFolder();
					if(!word.equals("") && !file.equals("") && !folder.equals(""))
					{
						makeArffFile  yeni=new makeArffFile();
						yeni.okuZemberekGonder(file, word,folder);
						JOptionPane.showInternalMessageDialog(pane, "Parsing is
completed.","Parse Completed", JOptionPane.INFORMATION_MESSAGE);
					}
					else if(word.equals(""))
					{
						JOptionPane.showMessageDialog(null, "Specified word can not be
found!", "ERROR", JOptionPane.ERROR_MESSAGE);
					}
					else if(file.equals(""))
					{
						JOptionPane.showMessageDialog(null, "No file selected!", "ERROR",
JOptionPane.ERROR_MESSAGE);
					}
					else if(folder.equals(""))
					{
						JOptionPane.showMessageDialog(null, "No path selected!", "ERROR",
JOptionPane.ERROR_MESSAGE);
					}
				}
			}
		}

		public void makeView()
		{
			pane = new JPanel();
			pane.setLayout(null);

			chooser = new JFileChooser();
			chooser2 = new JFileChooser();

			lbl_word = new JLabel("Specified Word:");
			lbl_word.setBounds(10, 10, 100, 20);
			pane.add(lbl_word);

			tf_word = new JTextField();
			tf_word.setBounds(110, 10, 120, 22);
			pane.add(tf_word);

			lbl_selectedFile = new JLabel("Select file of sentences:");
			lbl_selectedFile.setBounds(10, 35, 410, 20);
			pane.add(lbl_selectedFile);

			tf_selectedFile = new JTextField();
```

```
tf_selectedFile.setBounds(10, 57, 475, 22);
tf_selectedFile.setBackground(Color.WHITE);
//tf_selectedFile.setEditable(false);
tf_selectedFile.setText(".//denebas.txt");
pane.add(tf_selectedFile);

btn_selectedFile = new JButton("Select File");
btn_selectedFile.setBounds(10, 85, 150, 22);
btn_selectedFile.setFocusable(false);
btn_selectedFile.addActionListener(new Actions());
pane.add(btn_selectedFile);

btn_clearFile = new JButton("Clear Selection");
btn_clearFile.setBounds(165, 85, 150, 22);
btn_clearFile.setFocusable(false);
btn_clearFile.addActionListener(new Actions());
pane.add(btn_clearFile);

lbl_selectedFolder = new JLabel("Selected Path For ZemberekLast Creation:");
lbl_selectedFolder.setBounds(10, 115, 410, 20);
pane.add(lbl_selectedFolder);

tf_selectedFolder = new JTextField();
tf_selectedFolder.setBounds(10, 140, 475, 22);
tf_selectedFolder.setBackground(Color.WHITE);
//tf_selectedFolder.setEditable(false);
tf_selectedFolder.setText(".\\");
pane.add(tf_selectedFolder);

btn_selectedFolder = new JButton("Select Path");
btn_selectedFolder.setBounds(10, 168, 150, 22);
btn_selectedFolder.setFocusable(false);
btn_selectedFolder.addActionListener(new Actions());
pane.add(btn_selectedFolder);

btn_clearFolder = new JButton("Clear Selection");
btn_clearFolder.setBounds(165, 168, 150, 22);
btn_clearFolder.setFocusable(false);
btn_clearFolder.addActionListener(new Actions());
pane.add(btn_clearFolder);

lbl_operation = new JLabel("Operation:");
lbl_operation.setBounds(10, 198, 100, 20);
pane.add(lbl_operation);

btn_parse = new JButton("Parse");
btn_parse.setBounds(10, 223, 150, 22);
btn_parse.setFocusable(true);
btn_parse.addActionListener(new Actions());
pane.add(btn_parse);

btn_corpus = new JButton("Corpus");
btn_corpus.setBounds(165, 223, 150, 22);
btn_corpus.setFocusable(false);
btn_corpus.addActionListener(new Actions());
pane.add(btn_corpus);
```

```
	}

	public void makeFrame(){
		frm = new JFrame("Zemberek Parser");
		frm.setContentPane(pane);
		frm.setSize(500,300);
		Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
		int w = frm.getWidth(), h = frm.getHeight() ;
		frm.setLocation(d.width/2 - w/2, d.height/2 - h/2);
		frm.setDefaultCloseOperation(frm.EXIT_ON_CLOSE);
		frm.setResizable(false);
		frm.setVisible(true);
	}

	public String getSelectedFile(){
		return tf_selectedFile.getText();
	}
	public String getSelectedFolder()
	{
		return tf_selectedFolder.getText();
	}
	public String getWord()
	{
		return tf_word.getText();
	}

	public static void main(String[] args) {
		anaProgram gui = new anaProgram();



	}

}
```

MakeArrfFile

```
import net.zemberek.erisim.Zemberek;
import net.zemberek.tr.yapi.TurkiyeTurkcesi;
import net.zemberek.yapi.Kelime;
import net.zemberek.yapi.KelimeTipi;
import net.zemberek.yapi.ek.Ek;

import java.util.*;
import java.io.*;

import javax.swing.JOptionPane;

import com.sun.xml.internal.ws.util.StringUtils;

public class makeArffFile {

	public String[][] data=new String[500][11];
	int cumleSay=0;
	String[] ektip;
```

```
Zemberek zemberek = new Zemberek(new TurkiyeTurkcesi());

KelimeTipi[] kelTip=KelimeTipi.values();

public void okuZemberekGonder(String path,String word,String folder)
{

        String cumle="";
        String regex="\\r|\\t|\\s+";
        try
        {
                BufferedReader oku = new BufferedReader(new FileReader(path));
                String kokYokCumle="Kök olmayan cümlelerin sýralarý\n";
        while(oku.ready())
   {
        cumle = oku.readLine();
                System.out.println("\ncumle:"+cumle) ;
           String[] kelimeDizi = cumle.split(regex);
           boolean kokVarmi=false;
           boolean kokVar2=false;
           String anlam=kelimeDizi[kelimeDizi.length-1];
                        anlam=anlam.substring(anlam.indexOf("{")+1,anlam.indexOf("}"));
                        String[] anlamDizi=anlam.split(",");
                        int anlamSayi=0;

           for(int i=0;i<kelimeDizi.length;i++)
                 {
                String  kelime = kelimeDizi[i];
                if(kelimeDizi.length-1 !=i)
                {
                        System.out.println("kelime1:"+kelime) ;
                                Kelime[] cozumler = zemberek.kelimeCozumle(kelime);
                                if(cozumler.length!=0)
                                {       //aranan kelime mi;
                                        for(int j=0;j<cozumler.length;j++)
                                        {
                                                if (cozumler[j].kok().icerik().equals(word.toLowerCase()))
                                                {       kokVarmi=true;kokVar2=true;     break; }
                                                else
                                                {       kokVarmi=false;}
                                        }
                                }
                                if(kokVarmi)
                                {
                                        //aranan kelime
                                        SetWordDetails(kelime,cumleSay,4,cumle);
                                        //2 önceki kelime
                                        if(i>1)
                                                SetWordDetails(kelimeDizi[i-2],cumleSay,0,cumle);
                                        else
                                        {
                                                data[cumleSay][0]="NULL";
                                                data[cumleSay][1]="NULL";
                                        }
                                        //bir önceki kelime
                                        if(i>0)
```

```
                                                SetWordDetails(kelimeDizi[i-1],cumleSay,2,cumle);
                                        else
                                        {
                                                data[cumleSay][2]="NULL";
                                                data[cumleSay][3]="NULL";
                                        }
                                        //bir sonraki kelime
                                        if(i<=kelimeDizi.length-2)
                                                SetWordDetails(kelimeDizi[i+1],cumleSay,6,cumle);
                                        else
                                        {
                                                data[cumleSay][6]="NULL";
                                                data[cumleSay][7]="NULL";
                                        }
                                        //2 sonraki kelime
                                        if(i<=kelimeDizi.length-3)
                                                SetWordDetails(kelimeDizi[i+2],cumleSay,8,cumle);
                                        else
                                        {
                                                data[cumleSay][8]="NULL";
                                                data[cumleSay][9]="NULL";
                                        }
                                        //data[cumleSay][10]=kelimeDizi[kelimeDizi.length-1].substring(1,
kelimeDizi[kelimeDizi.length-1].length()-1);
                                        data[cumleSay][10]=anlamDizi[anlamSayi];

                                System.out.println(data[cumleSay][0] + "," + data[cumleSay][1]
                                     + ","+data[cumleSay][2]+ ","+data[cumleSay][3]
                                     +"," + data[cumleSay][4] + "," + data[cumleSay][5]
                                             +"," + data[cumleSay][6] + "," + data[cumleSay][7]
                                             + ","+ data[cumleSay][8]+ ","+data[cumleSay][9]
                                             + ","+data[cumleSay][10]);
                                        cumleSay++;
                                        kokVarmi=false;
                                        anlamSayi++;
                                }//kokvarmi
                        }//kelimeDizi.length
                         }//for
                if(!kokVar2)
                     kokYokCumle+=cumle+"\n";
           }//while
         oku.close();
         WriteToFile(folder + "\\result.arff");
         System.out.println(kokYokCumle);
        }catch (IOException e){
                System.out.println(e);
        }
        }
        public void SetWordDetails( String kelime,int sira,int atrribSira,String cumle)
        {
                Kelime[] cozumler = zemberek.kelimeCozumle(kelime);
                        if(cozumler.length!=0)
                        {
                                //çözüm seçme;
                           if(cozumler.length==1)
                                {
```

```
				data[sira][atrribSira]=cozumler[0].kok().tip().name();
				data[sira][atrribSira+1]=GetFirstAffix(cozumler[0]);
				}
				else
				{
					String ax=cumle+" cümlesindeki '"+kelime+"' kelimesi için uygun çözümü
seçiniz\n";
					for(int m=1;m<=cozumler.length;m++)
					{
						ax+=m+":"+cozumler[m-1]+"\n";
					}
					ax+="0:Diðer\n";
					String secim=null;
					while (secim==null)
					{
						secim= JOptionPane.showInputDialog(null,ax,"1");
						if(secim!=null)
						{
							if(Character.isDigit(secim.charAt(0)))
							{
								int secSayi=Integer.parseInt(secim);
								/*for(int m=1;m<=cozumler.length;m++)
								{
									if(secSayi==m)
									{	secim=String.valueOf(secSayi);break;}
									else
										secim=null;
								}*/
								if(secSayi>cozumler.length)
									secim=null;
							}
							else secim=null;
						}
					}
					if(secim!=null)
					{
						int sonuc=Integer.parseInt(secim);
						if(sonuc==0)
						{
							String tiplistKel=kelime+" kelimesi için uygun kök tipini
seçiniz\n";
							tiplistKel+=kokTipGetir();
							int
sonucKoktip=Integer.parseInt(JOptionPane.showInputDialog(null,tiplistKel,"1"));
							data[sira][atrribSira]=kelTip[sonucKoktip].toString();

							String tiplistEk=kelime+" kelimesi için uygun 1.ek tipini
seçiniz\n";
							tiplistEk+=ekTipGetir();
							int
sonucEktip=Integer.parseInt(JOptionPane.showInputDialog(null,tiplistEk,"1"));
							data[sira][atrribSira+1]=ektip[sonucEktip];
						}
						else{
							data[sira][atrribSira]=cozumler[sonuc-1].kok().tip().name();
							data[sira][atrribSira+1]=GetFirstAffix(cozumler[sonuc-1]);
```

```
                                    }
                              }
                        }
                  }
                  else
                  {
                        data[sira][atrribSira]="NULL";
                        data[sira][atrribSira+1]="NULL";
                  }

      }
      public String GetFirstAffix(Kelime kelime)
      {
            String affix="";
            // Eðer kelimenin eki varsa
            if(kelime.gercekEkYok())
                  affix = "NULL";
            else
            {
                  List<Ek> ekler=kelime.ekler();
                  affix=ekler.get(1).ad();
            }
            return affix;
      }
      public void WriteToFile(String path)
      {
            try {
          BufferedWriter out = new BufferedWriter(new FileWriter(path));

          PrepareAttributes(out);
          PrepareData(out);

          out.close();
        } catch (IOException e) {
        }

      }

      public void PrepareAttributes(BufferedWriter out) throws IOException
      {
            out.write("\r\n");
            out.write(" @relation meanings-of-words ");
            out.write("\r\n\r\n");

            String[] sortedArray = new String[cumleSay];

            for(int j=0; j<11; j++)
            {

                  // Result arrayini sort etmek için geçici bir array'e kopyalýyoruz
                  //System.out.print("\ncumle say:"+cumleSay+",data len:"+data[0].length);
            /*    for(int i=0; i<cumleSay; i++)
                  {
                        sortedArray[i] = data[i][j];

                  }*/
```

```
//      Arrays.sort(sortedArray);

        if(j == 0) out.write(" @attribute iki-önceki-kelime-tipi\t\t  ");
        else if(j == 1) out.write(" @attribute iki-önceki-kelime-ilk-eki\t  ");
        else if(j == 2) out.write(" @attribute önceki-kelime-tipi\t\t  ");
        else if(j == 3) out.write(" @attribute önceki-kelime-ilk-eki\t\t  ");
        else if(j == 4) out.write(" @attribute anlami-aranan-kelime-tipi\t  ");
        else if(j == 5) out.write(" @attribute anlami-aranan-kelime-ilk-eki\t  ");
        else if(j == 6) out.write(" @attribute sonraki-kelime-tipi\t\t  ");
        else if(j == 7) out.write(" @attribute sonraki-kelime-ilk-eki\t\t  ");
        else if(j == 8) out.write(" @attribute iki-sonraki-kelime-tipi\t\t  ");
        else if(j == 9) out.write(" @attribute iki-sonraki-kelime-ilk-eki\t  ");
        else if(j == 10) out.write("@attribute anlam\t\t\t\t\t  ");


        // attribute'lerin alabileceði deðerleri tek tek yazar
        out.write("{"+data[0][j] );
        //sortedArray[0]=data[0][j];
        int sortSira=1;
        boolean nullvar=false;
        for(int i=1; i < cumleSay; i++)
        {       int esitVar=0;
                for(int k=0;k<i;k++)
                {
                        if(data[i][j]==null )
                        {       if(nullvar)
                                {  esitVar=1;
                                        break;}
                                else {nullvar=true; esitVar=0;}
                        }
                        else if(data[i][j].equals(data[k][j]))
                        {   esitVar=1;
                                break;
                        }
                }
                if(esitVar==0)
                        out.write("," + data[i][j]);
        }
        out.write("}\r\n");

        /*out.write("{" + sortedArray[0]);
        System.out.print("{" + sortedArray[0]);
        for(int i=1; i < cumleSay; i++){
                if(!sortedArray[i].equals(sortedArray[i-1]))
                {
                        out.write("," + sortedArray[i]);
                        System.out.print("," + sortedArray[i]);
                }
        }
        out.write("}\r\n");
        System.out.print("}\r\n");*/
}

out.write("\r\n");

}
```

```
	public void PrepareData(BufferedWriter out) throws IOException
	{
		out.write(" @data\r\n %\r\n % " + cumleSay + " instance(s) \r\n %\r\n");

		// tüm alýnan datalarý yazar
		for(int i=0; i < cumleSay; i++)
      {
	out.write(" " + data[i][0] + "," + data[i][1] + ","); // Ýki önceki kelimenin tipi ve ilk eki
	out.write(" " + data[i][2] + "," + data[i][3] + ","); // Önceki kelimenin tipi ve ilk eki
	out.write(" " + data[i][4] + "," + data[i][5] + ","); // Aranan kelimenin tipi ve ilk eki
	out.write(" " + data[i][6] + "," + data[i][7] + ","); // Sonraki kelimenin tipi ve ilk eki
	out.write(" " + data[i][8] + "," + data[i][9] + ","); // Ýki sonraki kelimenin tipi ve ilk eki
	out.write(" " + data[i][10] + "\r\n");
      }
	}

	public String ekTipGetir()
	{
		String son="";
		String
ekler="NULL,ISIM_COGUL_LER,ISIM_KUCULTME_CIK,ISIM_KUCULTME_CEGIZ,ISIM_YONELM
E_E,ISIM_KALMA_DE,ISIM_CIKMA_DEN,ISIM_TANIMLAMA_DIR,ISIM_BELIRTME_I,ISIM_DUR
UM_LIK,ISIM_GIBI_CE,ISIM_ANDIRMA_IMSI,ISIM_ANDIRMA_SI,ISIM_TAMLAMA_IN,ISIM_TA
MLAMA_I,ISIM_BULUNMA_KI,ISIM_TARAFINDAN_CE,ISIM_DONUSUM_LE,ISIM_DONUSUM_L
ES,ISIM_BULUNMA_LI,ISIM_BULUNMA_LIK,ISIM_BIRLIKTELIK_LE,ISIM_SAHIPLIK_BEN_IM,I
SIM_SAHIPLIK_SEN_IN,ISIM_SAHIPLIK_O_I,ISIM_SAHIPLIK_BIZ_IMIZ,ISIM_SAHIPLIK_SIZ_INI
Z,ISIM_SAHIPLIK_ONLAR_LERI,ISIM_ILISKILI_SEL,ISIM_YOKLUK_SIZ,ISIM_ILGI_CI,ISIM_KISI
_BEN_IM,ISIM_KISI_SEN_SIN,ISIM_KISI_O_BOS,ISIM_KISI_BIZ_IZ,ISIM_KISI_SIZ_SINIZ,ISIM_KI
SI_ONLAR_LER,SAYI_ULESTIRME_ER,SAYI_KESIR_DE,SAYI_SIRA_INCI,SAYI_TOPLULUK_IZ,S
AYI_KOSE_GEN,FIIL_MASTAR_CE,FIIL_GIBI_CESINE,FIIL_DEVAMLILIK_DIKCE,FIIL_DONUSU
M_EN,FIIL_DONUSUM_IS,FIIL_DONUSUM_IK,FIIL_DONUSUM_IM,FIIL_DONUSUM_ILI,FIIL_DO
NUSUM_INTI,FIIL_DONUSUM_ESICE,FIIL_DONUSUM_ESI,FIIL_DONUSUM_ESIYE,FIIL_DONUS
UM_ME,FIIL_DONUSUM_MEZ,FIIL_DONUSUM_ECEK,FIIL_DONUSUM_MIS,FIIL_TANIMLAMA_I
CI,FIIL_EDILGEN_IL,FIIL_EDILGENSESLI_N,FIIL_BERI_ELI,FIIL_EMIR_O_SIN,FIIL_EMIR_ONLA
R_SINLER,FIIL_EMIR_SIZ_IN,FIIL_EMIR_SIZRESMI_INIZ,FIIL_SUREKLILIK_EREK,FIIL_ETTIRG
EN_TIR,FIIL_ETTIRGEN_TEKRAR_T,FIIL_GELECEKZAMAN_ECEK,FIIL_GECMISZAMAN_DI,FIIL
_GECMISZAMAN_MIS,FIIL_GENISZAMAN_IR,FIIL_SIMDIKIZAMAN_IYOR,FIIL_ZAMAN_INCE,F
IIL_IMSI_IP,FIIL_BERABERLIK_IS,FIIL_ISTEK_E,FIIL_MASTAR_MEK,FIIL_OLDURGAN_T,FIIL_O
LUMSUZLUK_DEN,FIIL_OLUMSUZLUK_ME,FIIL_YETERSIZLIK_E,FIIL_BELIRTME_DIK,FIIL_KI
SI_BEN,FIIL_KISI_BIZ,FIIL_KISI_O,FIIL_KISI_ONLAR,FIIL_KISI_SEN,FIIL_KISI_SIZ,FIIL_SART_S
E,FIIL_ISTEK_SENE,FIIL_ISTEK_SENIZE,FIIL_OLUMSUZLUK_SIZIN,FIIL_SURERLIK_EDUR,FIIL
_SURERLIK_EGEL,FIIL_SURERLIK_EGOR,FIIL_SURERLIK_EKAL,FIIL_TEZLIK_IVER,FIIL_YAKL
ASMA_AYAZ,FIIL_YETENEK_EBIL,FIIL_ZORUNLULUK_MELI,YANKI_DONUSUM_TI,YANKI_D
ONUSUM_DA,ZAMAN_BELIRTME_KI,ZAMIR_SAHIPLIK_IN,ZAMIR_SAHIPLIK_IM,IMEK_RIVAY
ET_MIS,IMEK_SART_SE,IMEK_HIKAYE_DI,IMEK_ZAMAN_KEN,GENEL_HAL_E,GENEL_HAL_I,
GENEL_HAL_TAMLAMA_I,IMEK_ONLAR_LER,IMEK_ONLAR_LER,FIIL_SORU_MI";
		ektip=ekler.split(",");
		for (int k=0; k<ektip.length;k++) {
			if(k%5==0)
				son+="\n";
			son+=k+"-"+ektip[k]+"     \t";
		}
		return son;
	}
	public String kokTipGetir()
```

```
        {
                String son="";
                for (int k=0; k<kelTip.length;k++) {
                        son+=k+"-"+kelTip[k]+"\n";
                }
                return son;
        }
}
```