LING BROWSER
A NLP BASED BROWSER
FOR
LINGUISTIC INFORMATION

by
ÖNSEL ARMAĞAN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
February 2008

to my parents

Acknowledgements

I wish to express special thanks to my supervisor Kemal Oflazer, who has supported me in several ways in this project. His motivation and encouragement will always guide me through out my professional career.

LING BROWSER – A NLP BASED BROWSER FOR LINGUISTIC INFORMATION

Önsel ARMAĞAN

Computer Science and Engineering, Master of Science Thesis, 2008

Thesis Supervisor: Prof. Kemal OFLAZER

Keywords: Natural Language Processing, Computer Assisted Language Learning, Morphological Analysis

Abstract

Linguistic students and researchers need practical tools providing information about elements of a language to understand its properties and conduct research on that language. Many computer assisted language learning tools have been developed since the emerging of computers. However, none of these tools aim to satisfy the needs of advanced learners. In this thesis, we introduce a tool, LingBrowser, which is an intelligent hyper-text browser that employs natural language processing technology to provide an interactive environment for advanced language learners to access all kinds of linguistic information about the words in a Turkish text. LingBrowser provides immediate information about morphological, segmental, pronunciation and semantic properties about the words in any text. Also, with a search interface, LingBrowser can locate examples of many linguistic phonemena in the source text.

LING BROWSER – DİLBİLİM BİLGİSİ İÇİN NLP TABANLI TARAYICI

Önsel ARMAĞAN

Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi, 2008

Tez Danışmanı: Prof. Kemal OFLAZER

Anahtar Sözcükler: Doğal Dil İşleme, Bilgisayar Destekli Dil Öğrenimi, Biçimbilimsel Çözümleme

Özet

Dilbilim öğrencileri ve araştırmacıları, bir dilin özelliklerini anlamak ve üzerinde çalışma yapmak için o dilin öğeleri hakkında bilgi sağlayan kullanışlı araçlara ihtiyaç duymaktadırlar. Bilgisayarların ortaya çıkışından beri pek çok bilgisayar destekli dil öğrenim aracı geliştirilmiştir. Ama bu araçlardan hiçbiri ileri düzeydeki kullanıcıların ihtiyaçlarını karşılayacak şekilde düzenlenmemiştir. Bu tezde, ileri düzey dil öğrencileri için Türkçe bir metinde bulunan sözcüklerle ilgili her türlü dilbilimsel bilgiyi elde edebilecekleri etkileşimli bir ortam sağlayacak şekilde doğal dil işleme teknolojileri kullanan, LingBrowser adını verdiğimiz akıllı yardımlı metin tarayıcıyı tanıtmaktayız. LingBrowser herhangi bir metin içinde seçilen sözcüklerin biçimbilim, söyleniş, anlambilim özellikleri hakkında anında bilgi vermektedir. Aynı zamanda, bir arama arabirimi yoluyla, bir dil olayının metin içindeki tüm örneklerinin yerlerini belirlemek de mümkündür.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1
INTRODUCTION

1.1 Motivation

The primary goal of teaching linguistics is not that students memorize the linguistic rules of a language, but rather that they comprehend to recognize the structures where these rules are used in. Without practical exercise and training, it can be hard to achieve this goal. Also, it is generally accepted that hands-on experience motivates people to carry out research and stimulates thinking as research continues. As a result, linguistics students or researchers who wish to understand linguistic properties of a language, and conduct research on that language, need practical tools and resources. This need is more critical for Turkish since it has complex word structures that makes the linguistic exploration more difficult.

One general way of computationally implementing a training tool is to put all previous self-training materials like exercises, drills, explanations into electronic form and repeatedly propose exercises to learners. These exercises, however, can have limited feedback to the learner that will often be like ”the answer is correct” or ”the answer is wrong”. When acquiring a language, but not just learning the basics, then we need more communicative tools. For example, let us imagine a linguistics student encountering an unknown word or an unfamiliar usage of a known word while reading an external text. In this situation, the linguistic student will desire to learn the meaning, structure of the word etc. Furthermore, she may need to find examples of similarly structured words. These can be done only by using computational language techniques.

In this thesis, we introduce a tool, LingBrowser, that have a different approach to aid advanced learners while reading. LingBrowser is an intelligent hyper-text browser that employs natural language processing technology to provide an interactive environment for advanced language learners to access all kinds of linguistic information about the words in a Turkish text. It is interactive because users of LingBrowser can interact with a Turkish text in many ways by requesting information about properties of words and receiving answers and explanations.

LingBrowser can make non-computational explanations of linguistic phenomena available from the underlying computational representations.

1.2 Overview of LingBrowser Functionality

LingBrowser provides information about morphological segmentation and features, alignments of lexical and surface morphemes along with the explanation of any allomorph, segmental structure, pronunciation and any related explanations about pronunciation phonemena like location of stress in a word. By using WordNet [2, 3], which is a concept ontology database, meanings of the root can be accessed that one can observe the semantic properties of a word. Users can locate examples of many linguistic phenomena in the source text by performing search with various criteria.

1.3 Overview of Implementation

Reading has a crucial part in language learning that it consolidates previously learned material, increases the knowledge of vocabulary and more importantly it provides a relaxed, tension-free learning environment. The largest source of reading material that can be reached through computers is the Internet. Thus, we desired the LingBrowser to be a tool that aids language learners while they are browsing the web pages which are in HTML (Hyper Text Markup Language) format. With this idea in mind, we converted the popular web browser, Mozilla-Firefox, into a language learning environment with an add-on.

Mozilla-Firefox add-on development framework has programming limitations when you try to implement a complex application. For instance, there is no interface for database connectivity. On the other hand, it provides interfaces for data exchange in XML(eXtended Markup Language) and HTML (Hyper Text Markup Language) from external resources. Thus, we implemented a server application which does the most of computation and the add-on will be responsible for just simple manipulations and presentation of processed data.

1.4 Layout of the Thesis

The organization of thesis as follows: Chapter 2 introduces NLP Technologies and the relation of CALL (Computer Assisted Language Learning) and NLP. Chapter 3 presents the functionality of LingBrowser. Chapter 4 discusses the issues of implementation. In Chapter 5 we will present a session with LingBrowser. Finally we will conclude the thesis in Chapter 6.

Chapter 2
NATURAL LANGUAGE PROCESSING TECHNOLOGY

2.1 General Notions

In this chapter, we will focus on NLP technology for analysis of words and applications of NLP in computer assisted language learning. We will start with some brief information about NLP and definitions of linguistics terms.

NATURAL LANGUAGES, including Turkish, English, Arabic etc., are the written and spoken communication systems between human beings. *NATURAL LANGUAGE PROCESSING* (NLP), in a broad term, tries to convert *Natural Languages* into formal representations that they can be analyzed or generated by computers. Some of applications of NLP can be listed as machine translation, automatic summarization, question answering, etc.

LINGUISTICS is a wide field that studies natural languages, and *MORPHOLOGY* is the branch of linguistics that deals with words. In most languages words can have complex grammatical structures. A word is composed of *MORPHEMES* which are the smallest units of structure. Morphemes can express either morphosyntactics or semantics. The morphemes which express the semantic features are *ROOT*s or *STEM*s. A word will generally have one root morpheme and multiple affix morphemes added to this morpheme.

Affixes that appear before root are called *PREFIX* and affixes that appear after root are called *SUFFIX*. In *AGGLUNATIVE* languages, like Turkish, affixes are attached to roots sequentially like "beads on a string".

2.2 Morphology

2.2.1 Challenges Of Morphology

There are two main challenges in morphology for linguists and NLP practitioners.

1. Morphotactics

2. Morphophonology

Morphotactics

Morphotactics is the study of how valid words are constructed. Thus, one should define all the possibilities and limitations of word constructions to do a morphological analysis.

In a natural language, how the construction of word out of morphemes can occur depends on the morphotactic rules of that language. The most common way of constructing a word is simply by concatenating morphemes; however many constraints affect concatenations. As an example, In English suffix `+ation` can be attached to only verbs and produces nouns (`compute` + `ation` → `computation`).

Morphophonology

Morphophonology is the study of alternations during the word construction. One can assume that knowledge of morphotactics can be enough to break down a word into morphemes, however there is another aspect of natural language which makes things more complicated. Through the process of combination of morphemes, some alternations may appear in form of morphemes. These alteration can appear as assimilation of phonemes, deletion of phonemes or introduction of new phonemes. As an example, In English, when we add the plural morpheme `+s` to the root morpheme `leaf`, the word constructed will be `leaves`. In this case, `f` is assimilated to `v` and a new phoneme `e` is introduced.

2.2.2 Turkish Morphology

Turkish is a agglunative language in which words consist of morphemes which are concatenated to root morpheme as beads on a string. Turkish words generally has one root morpheme and two or more suffixes. Each morpheme produces an inflection (a change in grammatical information such as tense, number, person, case, etc.) or a derivation (a change in syntactic group of word such as a change from a verb to a noun). Thus, multiple inflections or derivations may occur in a Turkish word which makes the understanding of morphotactics of the word more tricky for a non-native speaker. As an example, surface morphemes of the Turkish word `kolaylaştırdığım` (*‘ that which I caused it to become easy’*) is separated into morphemes as:

kolay+laş+tır+dığ+ım

The adjective root `kolay` meaning *‘easy’*, is derived into a verb meaning *to become easy* with morpheme `+laş` . The next surface morpheme `+tır` is a causative morpheme that changes the meaning to *‘to cause to become easy’*. Then, past participle marker `+dığ` and 1st person singular possessor `+ım` produce the final form of the word, meaning roughly (*' that which I caused it to become easy'*).

Another complexity of Turkish is that surface representations of morphemes are conditioned by various morphophonemic process such as vowel harmony, consonant assimilations and elisions. As a result, a lexical morpheme can be realized in multiple surface forms. For the above example, the lexical morpheme structure is:

```
kolay+lAş+DHr+DHk+Hm
```

In this form, some meta characters which correspond to a set of graphemes are employed to represent lexical morphemes. For instance, `H` is used for high vowels, (`u,ü,ı,i`) and `D` is used for alveolar consonants (`d , t`) in orthography. Thus, lexical morpheme `+DHr` can represent 8 different allomorphs (`+tir, +tır, +tur, +tür, +dır, +dir, +dur, +dür`) according to the context.

2.2.3 Morphological Analyzer

Main objective of a morphological analyzer is to abstract away the morphotactic and morphophonological process, and to break down the word into its component morphemes. When we consider the morphotactics of a language as simple concatenations, then it seems that structure of the word can simply be described as finite automata. However, in finite automata, the description of morphophonological process was not clear until the introduction of two-level morphology by Koskenniemi [5]. The system proposed by Koskenniemi enables linguists to use finite state transducers to handle especially morphophonological processes. After the introduction of two-level morphology, morphological analyzers are implemented for many languages. In this thesis, we are going to make use of the morphological analyzer for Turkish designed by Oflazer [1].

Once the morphemes are located in a word, one can then map these morphemes to other morphological features like lexical features, pronunciation, etc., and obtain further information about the word such as location of stress mark. For the above we can construct following mappings:

```
kolay → kolay+Adj
lAş → ^DB+Verb+Become
DHr → ^DB+Verb+Caus+Pos
DHk → ^DB+Adj+PastPart
Hm  →  P1sg
```

In these mappings, `DB` stands for *derivation boundary.* When we replace the lexical morphemes with these mappings, we get the morphological feature analysis of the word. In this thesis, we call this resulting representations simply as the morphological analysis.

```
kolay+Adj^DB+Verb+Become^DB+Verb+Caus+Pos^DB+Adj+PastPart+P1sg
```

Note that these mappings are not one-to-one, thus morphological analysis of a word may have multiple results. A morphological analyzer should present all the possible results.

2.3 Other Issues of Natural Language Processing

Morphological analysis is the first step in NLP. When we try to process group of words, more issues arise. Some of these can be listed as:

- *Text Segmentation:* Before processing a text, it needs to be segmented to its smaller units, such as words, punctuations, numbers, etc.

- *Parsing:* Some strings of the words should be assigned to a syntactic analysis like noun phrases, adjective phrases.

- *Word-Sense Disambiguation:* Some words can have multiple meanings. Correct meaning of the word in a text should be determined.

NLP does not only deal with written language, but also spoken language. Some topics of processing of spoken language can be listed as:

- Speech Recognition

- Text-To-Speech Synthesis

2.4 Natural Language Processing in Computer Assisted Language Learning

The idea of using computers in language learning is not new and many computer assisted language learning(CALL) applications have been developed since the 1960s. However, NLP use has been very limited in these applications. Nerbonne [6], in his survey of NLP usage in CALL applications, clearly states that only a few of the CALL applications utilize the techniques of NLP. Also, NLP practitioners do not see CALL as an interesting research area. Borin [8], in a recent paper that reviews the relation between these two research areas, states that CALL does not seem to have a place in natural language processing. Essentially, Borin concludes that

> ”... in the eye of casual beholder - the two disciplines seem to live completely different words.”.

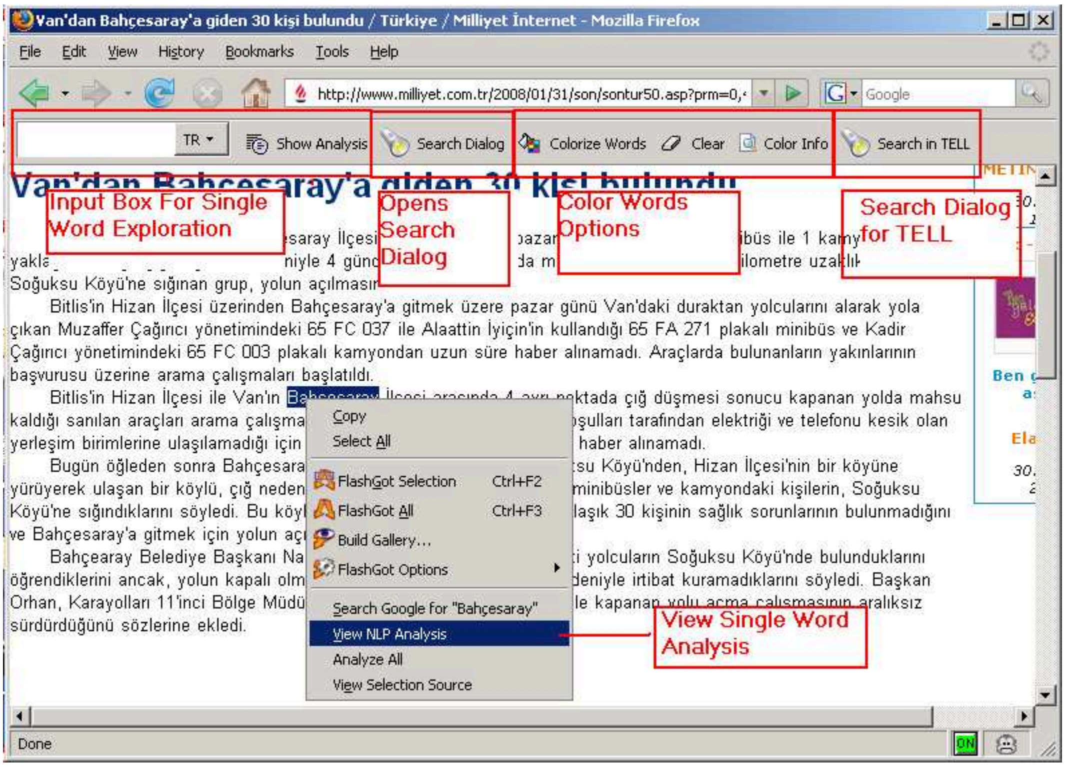
Despite these views, recently a number of projects that make use of natural language processing techniques in language learning have emerged. One of the most successful application is GLOSSER project [7]. The Glosser Project has developed a system that helps the readers of foreign language by providing access to a dictionary after a morphological analysis and part-of-speech disambiguation of the word. For Turkish, we can cite the works done by Güvenir [9] and, Güvenir and Oflazer [10]. These two works introduce a corpus based tutoring system where the corpus is composed of sentences collected by authors. They proposed a system where users can search the correct usage of various grammatical rules in this corpus.

However, none of these works aim to satisfy the needs of advanced learners such as linguistics students.

Chapter 3
FUNCTIONALITY OF LINGBROWSER

In this chapter, we will introduce the functionality of LingBrowser in three main groups: Single Word Exploration, Search Functionality and Additional Features.

The main window of LingBrowser is shown in Figure 3.1 where one can load HTML file. All the functionality is available with a tool-bar, right-click menu and mouse double-click option.



Van'dan Bahçesaray'a giden 30 kişi bulundu

saray İlçesi pazar ibüs ile 1 kamy
yakla niyle 4 günc da m ilometre uzaklık
Soğuksu Köyü'ne sığınan grup, yolun açılmasın

Bitlis'in Hizan İlçesi üzerinden Bahçesaray'a gitmek üzere pazar günü Van'daki duraktan yolcularını alarak yola çıkan Muzaffer Çağırıcı yönetimindeki 65 FC 037 ile Alaattin İyiçin'in kullandığı 65 FA 271 plakalı minibüs ve Kadir Çağırıcı yönetimindeki 65 FC 003 plakalı kamyondan uzun süre haber alınamadı. Araçlarda bulunanların yakınlarının başvurusu üzerine arama çalışmaları başlatıldı.

Bitlis'in Hizan İlçesi ile Van'ın Bahçesaray İlçesi arasında 4 ayrı noktada çığ düşmesi sonucu kapanan yolda mahsu
kaldığı sanılan araçları arama çalışma oşulları tarafından elektriği ve telefonu kesik olan
yerleşim birimlerine ulaşılamadığı için haber alınamadı.

Bugün öğleden sonra Bahçesara su Köyü'nden, Hizan İlçesi'nin bir köyüne
yürüyerek ulaşan bir köylü, çığ neden minibüsler ve kamyondaki kişilerin, Soğuksu
Köyü'ne sığındıklarını söyledi. Bu köyl laşık 30 kişinin sağlık sorunlarının bulunmadığını
ve Bahçesaray'a gitmek için yolun aç

Bahçearay Belediye Başkanı Na i yolcuların Soğuksu Köyü'nde bulunduklarını
öğrendiklerini ancak, yolun kapalı olm deniyle irtibat kuramadıklarını söyledi. Başkan
Orhan, Karayolları 11'inci Bölge Müdü le kapanan yolu açma çalışmasının aralıksız
sürdürdüğünü sözlerine ekledi.

Figure 3.1: LingBrowser Main Window of LingBrowser

3.1 Single Word Exploration

Single word exploration is the part where one can access the morphological analysis, lexical and surface morpheme structure, pronunciation analysis and translation of a word. After loading an HTML or a text file into LingBrowser, if one double-clicks a word, enters a word in the input box of tool-bar or presses the View NLP Analysis item of right-click menu after highlighting the word, a pop-up window which includes these linguistic information about the selected word will appear. Following subsections illustrate the informations that are made available in the pop-up window.

3.1.1 Morphological Analysis

Morphological Analysis of a word consists of root of the word, part-of-speech tag (Noun, Verb, Adjective, ...) of the root, related inflectional and derivational features constructed by any suffixes. Inflectional features indicate grammatical informations about the word such as tense, person, number, case. On the other hand, derivational features change a word from one syntactic category to a different word from another syntactic category. For instance, in English, the derivational feature constructed by suffix `+ly` changes adjectives to adverbs (rapid ⇒ rapidly). Morphological analysis of a word can have multiple results if the queried word has multiple interpretations. For instance, LingBrowser displays two results for the word *kitabına*:

```
kitap+Noun+A3sg+P2sg+Dat
kitap+Noun+A3sg+P3sg+Dat
```

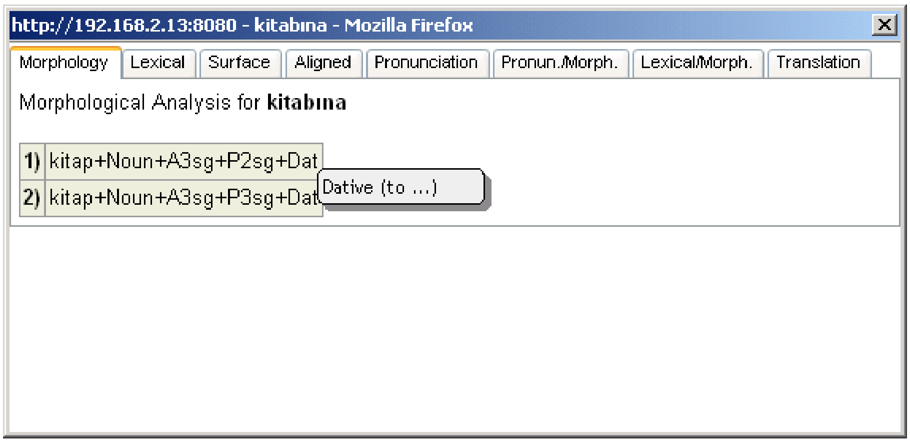
First analysis is the result of the interpretation ‘to your book’ and the second one corresponds to the interpretation ‘to his/her book’. In this example, we can see that *kitap* is the root of the word and part-of-speech tag of the root is *Noun*. Other parts are the inflectional features of the word. Following example which is the analysis of the word `iyilik` shows a derivation.

```
iyi+Adj^DB+Noun+Ness+A3sg+Pnon+Nom
```

As you notice, there is a part named as `DB` indicating that there is a derivation in the morphology of the word. For this case, we understand that the word *iyilik* (goodness) is a noun which is derived from an adjective root *iyi* (good) by a derivational feature `+Ness` [1].

[1] Roughly corresponding to the `+ness` in the English word `goodness`

As a further functionality, LingBrowser shows tool tips for each feature when mouse hovers on feature names. Since the feature names (`A3sg,Dat,P2sg, etc.`) in *Morphological Analysis* are encoded representations, one may need clarification for them. In the *kitabına* case, when the mouse hovers on `+Dat`, a tool tip which explains that the feature `+Dat` indicates *Dative Case* will be displayed. Figure 3.2 illustrates this functionality. (See Appendix A for the full list of features and their explanations)



Morphological Analysis for **kitabına**

1)	kitap+Noun+A3sg+P2sg+Dat
2)	kitap+Noun+A3sg+P3sg+Dat

Dative (to ...)

Figure 3.2: Morphological Analysis

3.1.2 Lexical Morpheme Structure

Lexical Morpheme Structure is the representation of morphemes of a word where the allomorphy caused by any morphographemic phonemena is abstracted away.

Abstracting the morpheme structure from these allomorphies is crucial for Turkish language, since a morpheme can evolve into different forms for different contexts. Fon instance, Turkish plural morpheme can appear as either `+ler` or `+lar` depending on vowel harmony. These allomorphies can easily confuse a non-native researcher while comparing the words. The next example will illustrate this importance. When we query the Turkish words, *kitabına* and *kedine*, through LingBrowser, the following results are displayed in Lexical Morpheme Structure tab of resulting window.

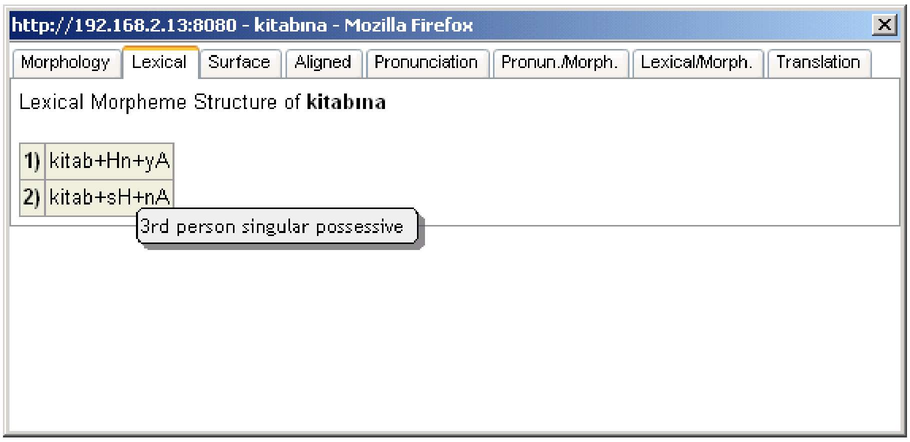
```
kitab+Hn+yA
kitab+sH+nA
```

and

kedi+Hn+yA

Although *kitabına* and *kedine* do not look similar, they have same lexical morpheme structures except for the roots. To eliminate the allomorphies in a morpheme, some meta-characters are employed to represent the groups of surface characters. In the above examples, H represents the high vowels(*ı,i,u,ü*) and A represents the non-round low vowels(*a,e*).

As an extended functionality, LingBrowser will display a description about the morphemes when the mouse hovers on them. For kitab+sH+nA case, when mouse hovers on morpheme +sH, *3rd Person Singular Possessive* will be displayed as a tool tip as one can see in Figure 3.3. (See Appendix D for all lexical morphemes.)



Lexical Morpheme Structure of **kitabına**

1) kitab+Hn+yA
2) kitab+sH+nA

3rd person singular possessive

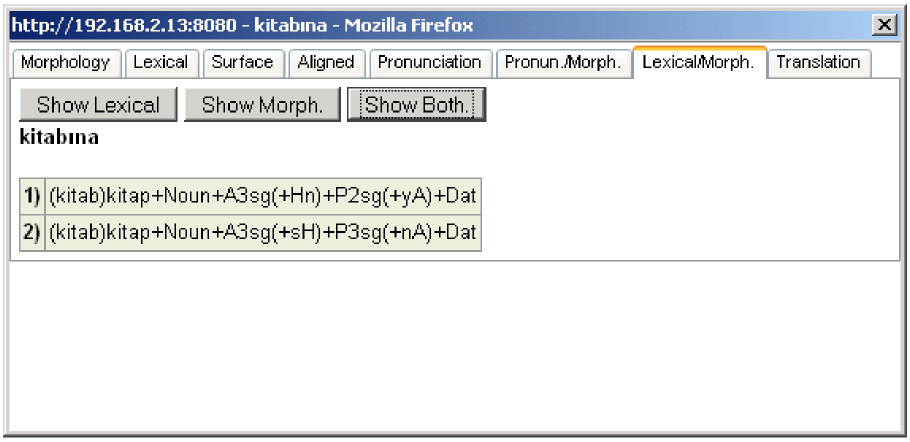
Figure 3.3: Lexical Morpheme Structure

3.1.3 Morphology and Lexical Morpheme Structure Alignment

In Section 3.1.1, we have covered the morphological analysis of a word which is the combination of features indicated by the morphemes of the word. However, one should see which features match which morpheme to have an understanding of underlying process. LingBrowser overcomes this issue by showing features and morphemes in an interleaved format. For the word *kitabına* LingBrowser displays feature and morpheme alignment as:

(kitab)kitap+Noun+A3sg(+Hn)+P2sg(+yA)+Dat
(kitab)kitap+Noun+A3sg(+sH)+P3sg(+nA)+Dat

In the first interpretation, one can see that the root *"kitab"* gives rise to features kitap, Noun and A3sg , morpheme Hn gives rise to feature P2sg, and last morpheme `yA` gives rise to feature `Dat`. Also, it is possible to hide either morphological features or lexical morphemes. Figure 3.4 shows how LingBrowser presents the information.



kitabına

1) (kitab)kitap+Noun+A3sg(+Hn)+P2sg(+yA)+Dat

2) (kitab)kitap+Noun+A3sg(+sH)+P3sg(+nA)+Dat

Figure 3.4: Alignment of Lexical Morphemes and Features

3.1.4 Surface Morpheme Structure

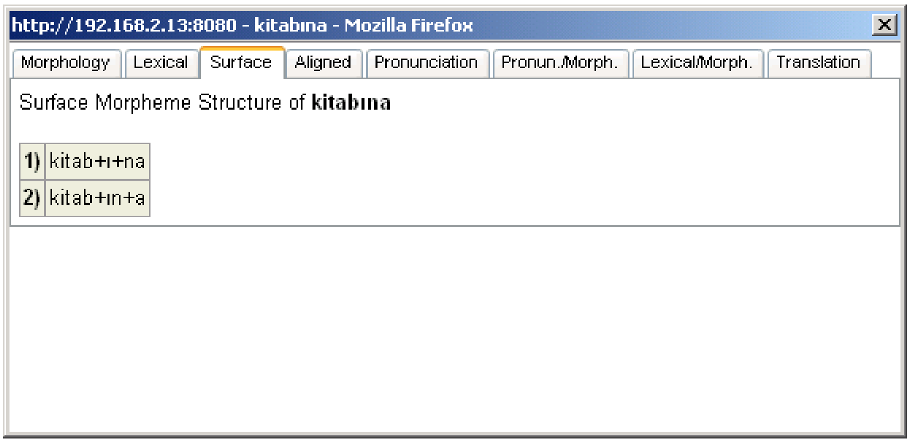
Surface Morpheme Structure of a word has the similar information as Lexical Morpheme Structure of a word has. However, all the meta-characters are converted to their corresponding surface phonemes and there may be possible deletions according to the morphology rules that applies. LingBrowser will display the following surface morpheme structure for word *kitabına*.

```
kitab+ın+a
kitab+ı+na
```

In the first interpretation, `H` is converted to `ı`, `A` is converted to `a` and `y` on the last morpheme is deleted because the previous morpheme ends with a consonant. Figure 3.5 is a screen shot for this subsection.

3.1.5 Lexical Surface Alignment

Morphological processes of a language involves the relations between surface morpheme structure and lexical morpheme structure, and the morphological rules that construct those relations. The *Lexical Surface Alignment* part is the one that gives ability to the user to observe the undergoing morphological process in the Turkish language. One can find which morphographemic rules are applied on the word



Surface Morpheme Structure of **kitabına**

1)	kitab+ı+na
2)	kitab+ın+a

Figure 3.5: Surface Morpheme Structure

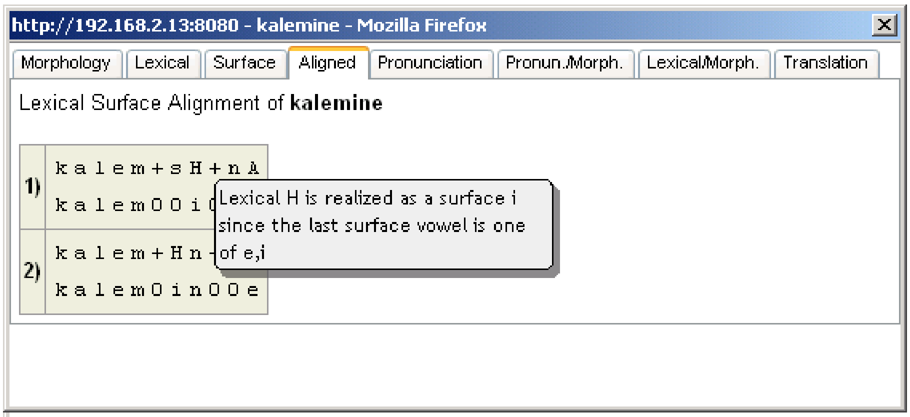
and monitor the results of those rules. As an example, LingBrowser shows the user the following lexical and surface alignment result when one queries the word *kalemine*.

```
kalem+sH+nA    kalem+Hn+yA
kalem00i0ne    kalem0in00e
```

One can see that the pair `(H,i)` is the result of morphographemic rule of Turkish vowel harmony. According to this rule `H` is realized as `i` because the last vowel before H is one of (e,i). If mouse hovers on one these pairs, LingBrowser displays explanation of mediating rule according to the two level grammar described in [1]. Figure 3.6 demonstrates this functionality. (See Appendix C for all feasible pairs and the explanation of rules that apply)

3.1.6 Pronunciation

Most of the Turkish words has only one pronunciation. However, some words that have multiple morphological interpretations may have multiple pronunciations. This usually happens when a loan word has a homograph of another Turkish word which has a different pronunciation. The difference may appear as a change in consonant quality or vowel length. Another source of ambiguity is the location of the primary lexical stress. In Turkish, the position of the primary stress usually depends on the stress marking properties of morphemes, but some root words may have exceptional stress. LingBrowser displays all the possible pronunciations



Lexical Surface Alignment of **kalemine**

1) k a l e m + s H + n A
k a l e m 0 0 i 0

2) k a l e m + H n +
k a l e m 0 i n 0 0 e

Lexical H is realized as a surface i since the last surface vowel is one of e,i

Figure 3.6: Alignment of Lexical Morphemes and Surface Morphemes

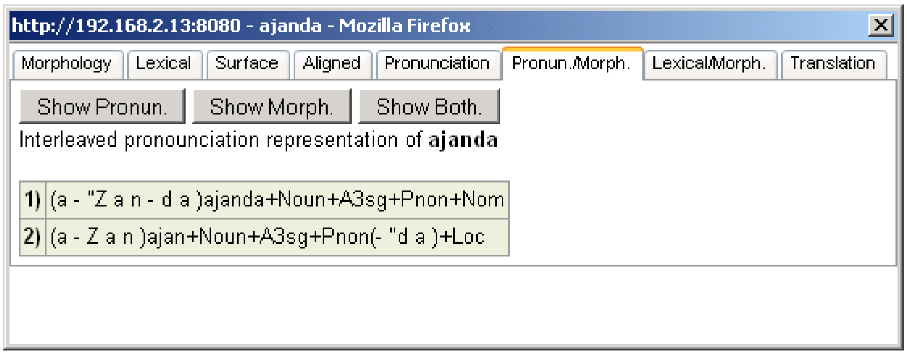
that a Turkish word may possess by using *SAMPA (Speech Assessment Methods Phonetic Alphabet)* and *IPA (International Phonetic Alphabet)*. For instance, when one look for the pronunciation of word `ajanda`, LingBrowser displays the following pronunciation which is in SAMPA format. (See Appendix C for the list of all SAMPA characters for Turkish pronunciation)

```
a - "Z a n - d a
a - Z a n - "d a
```

First pronunciation of the word corresponds to the interpretation *"agenda"* and the second one corresponds to the interpretation "on the agent". The two pronunciations differ in the position of stress mark (labeled as `"` in SAMPA). These results do not give any clue about the reason why these stress marks are located on those locations. Thus, LingBrowser also displays the morphological features and pronunciations that these features give rise to in an interleaved format. This representation looks like the following:

```
(a - "Z a n - d a )ajanda+Noun+A3sg+Pnon+Nom
(a - Z a n )ajan+Noun+A3sg+Pnon(- "d a )+Loc
```

In this representation, one can see that first interpretation has only one morpheme which is just the root. Now we can understand that first pronunciation has the stress mark on second syllable as a result of the exceptional stress property of the root. The second one has the stress on last syllable because where there are no exceptional stresses, stress is by default on the last syllable.(See the screen shot in Figure 3.7). Note that syllable and morpheme boundaries do not necessarily overlap. A morpheme can be split over multiple syllables or syllables may contain



Interleaved pronounciation representation of **ajanda**

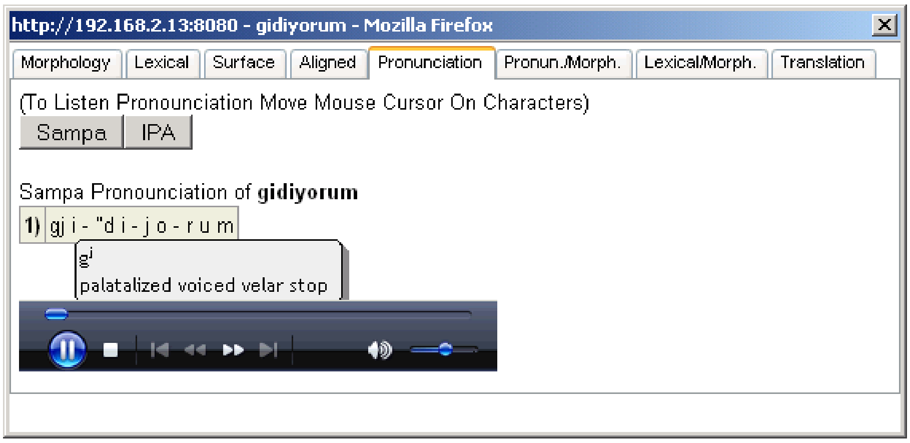
1)	(a - "Z a n - d a)ajanda+Noun+A3sg+Pnon+Nom
2)	(a - Z a n)ajan+Noun+A3sg+Pnon(- "d a)+Loc

Figure 3.7: Alignment of Features and Pronunciation

segments from multiple morphemes. To see an example, anlaysis of word *gidiyorum* can be explored. The surface morpheme `+iyor` has parts of different syllables and the syllable ‘ `-di-` spans over morphemes `git` and `+iyor`.

```
(gj i - "d )git+Verb+Pos(i - j o - r )+Prog1(u m )+A1sg
```

Another pronunciation functionality provided by LingBrowser is the ability to hear the pronunciation of single symbols in context. When mouse hovers on a character, the corresponding pronunciation is played and also a phonological description of that is shown in a tool tip. For instance, when mouse hovers on the `gj` for the above example, LingBrowser displays that the symbol denotes a *palatalized voice velar stop* phoneme. This functionality is exemplified in Figure 3.8.



(To Listen Pronounciation Move Mouse Cursor On Characters)

Sampa Pronounciation of **gidiyorum**

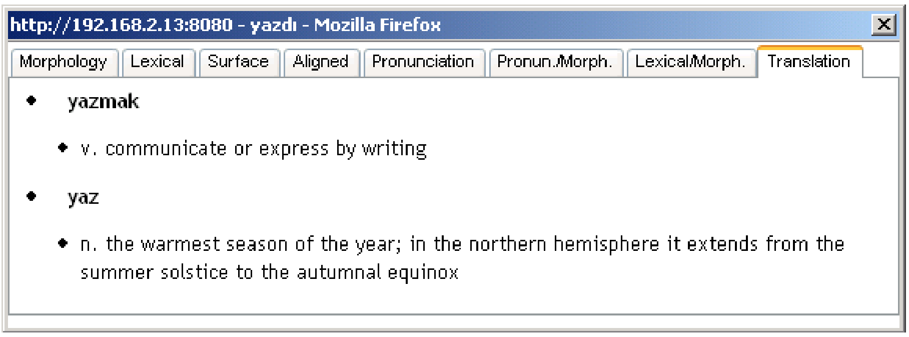
1) gj i - "d i - j o - r u m

g^j
palatalized voiced velar stop

Figure 3.8: Pronunciation of a Word

3.1.7 Word Translation

LingBrowser provides access to translations of the roots of the queried word from the glosses of English WordNet[2] using the interlingual-index number of the root which is obtained from Turkish WordNet [3]. The possible translations will be grouped according to the part of speech tagging of the root. For the Turkish word `yazdı`, the root word `yaz` has two possible interpretations. One is a verb (meaning ”to write”) and the other one is a noun (meaning ”summer”). The translation of these two interpretations will be presented in the pop-up window as in Figure 3.9.



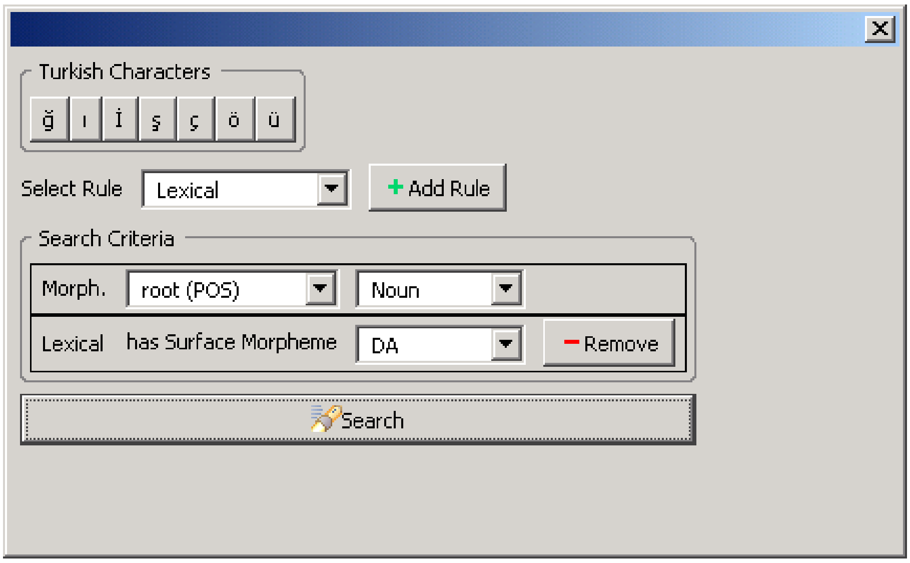
- yazmak
  - v. communicate or express by writing
- yaz
  - n. the warmest season of the year; in the northern hemisphere it extends from the summer solstice to the autumnal equinox

Figure 3.9: Translation of a Word

3.2 Search Engine

When working on linguistic property or a rule, it is always helpful to see the examples of applications of the rule for a better understanding. LingBrowser is able to locate words with various linguistic features through a search panel. In the search panel, one can define multiple rules and LingBrowser will locate the words which satisfy all the rules constructed by the user in the current text. The results of the search are presented in a result panel where one can reach the linguistic properties that are explained in Section 3.1. The search panel and the result panel can be seen in Figures 3.10 and 3.11.

There are four general category of the rules that can be constructed: *Morphology Rules, Lexical Morpheme Structure Rules, Ortography Rule and Pronunciation Rules*. Each category may have sub-categories which will be explained in the following subsections.



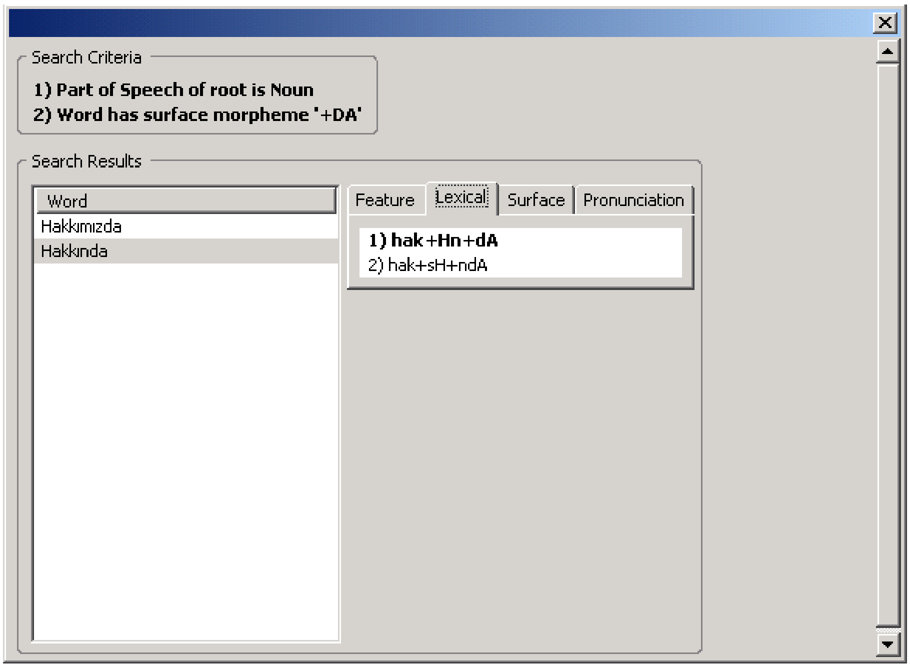
Select Rule: Lexical

Search Criteria

Morph. root (POS) Noun

Lexical has Surface Morpheme DA

Figure 3.10: Search Window



Search Criteria

1) Part of Speech of root is Noun
2) Word has surface morpheme '+DA'

Search Results

Word
Hakkımızda
Hakkında

1) hak+Hn+dA
2) hak+sH+ndA

Figure 3.11: Search Results Window

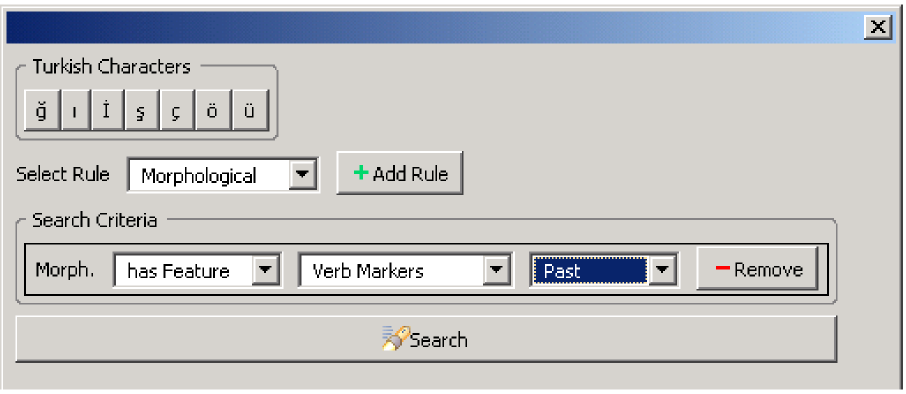
3.2.1 Morphology Rules

LingBrowser is able to search a Turkish text in order to locate the words that have the possession of various morphosyntactic features. Three kinds of rules can be defined in this part.

1. The first kind of rule is for searching words according to the part-of-speech tag of root (Verb, Noun, Adjective, etc). The possible part-of-speech tag can be chosen from drop down box. As an example, one may search words that have a verbal root.

2. Second kind of rule gives the user the ability to search words according to the orthography of the root. For example, the words with the root is `yaz` can be located by LingBrowser.

3. Third kind of rule is for searching words which contains a specific morphological feature. As an example, one may search the words with a past tense marker (`+Past`). The morphological features are grouped in categories. The possible morphological features that are filtered according to the chosen category will be available to user through a drop down box. Figure 3.12 demonstrate the usage of this rule.



Select Rule: Morphological

Morph. has Feature | Verb Markers | Past

Figure 3.12: Morphology Rules

These rules can be used individually and also they can be combined so that one can search words with the verbal root `yaz` and a past tense marker by defining multiple rules.

After defining rules and activating the search process LingBrowser will display the result in the search pane. Since a word may have multiple morphological interpretations, LingBrowser shows all the possible morphological interpretations and highlights the interpretation that meets the search criteria. If one searches a text for the words with a verbal root and past tense marker and if the word *yazdı* occurs in the text, LingBrowser will find this word and displays it in the result pane. If this word is selected in the result pane, morphological analysis of the word will be displayed as the following:

```
yaz+Verb+Pos+Past+A3sg
yaz+Noun+A3sg+Pnon+Nom^DB+Verb+Zero+Past+A3sg
```

In this case, since we are looking for a word with verbal root, only the first interpretation will be highlighted.

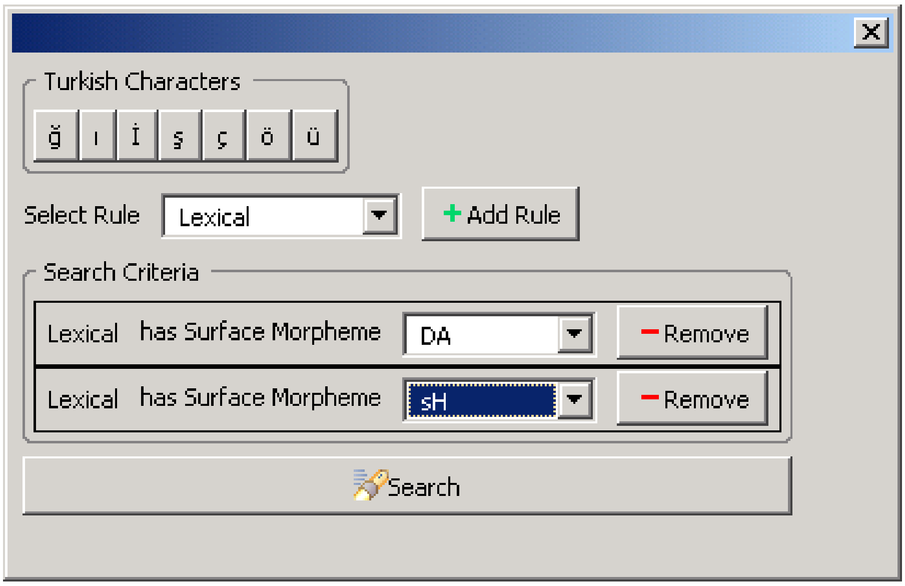
3.2.2 Lexical Morpheme Structure Rule

This rule is for locating words that have specified lexical morphemes in their lexical morpheme structure. Possible lexical morphemes is listed in a drop-down box and user will choose one of them to form a rule. If one defines multiple lexical rules, LingBrowser finds the words that meet the criteria of all rules. For instance, one may define two rules, one for morpheme `+DA` and one for morpheme `+sH`, the words that have both `+DA` and `+sH` in their morpheme structure will be located. In Figure 3.13, this property is exemplified. If one selects the word in the result pane, lexical morpheme structure analysis of the selected word will also be made available. Since, words may have multiple lexical morpheme structures LingBrowser will highlight the appropriate result.

3.2.3 Orthography Rule

By defining an orthography rule, one may search a Turkish text for words according to their surface forms. For instance, one may need to find the words that has `lik` at the end of the word or the words that starts with a sequence of letters *vowel+consonant+consonant+vowel* (e.g. `e l l i`).

LingBrowser employs simple regular expression rules and meta characters to achieve this goal. There is a help button which displays a guide for the usage of this rule. The meta characters that can be used in the construction of this are explained in Table 3.1. Table 3.2 lists some basic regular expressions containing these meta characters.



Select Rule Lexical

Lexical has Surface Morpheme DA

Lexical has Surface Morpheme sH

Figure 3.13: Lexical Morpheme Rule

Table 3.1: Orthography Meta Characters

Meta Character	Meaning
(A)	Any character used in Turkish orthography
(V)	Any character representing a vowel
(C)	Any character representing a consonant

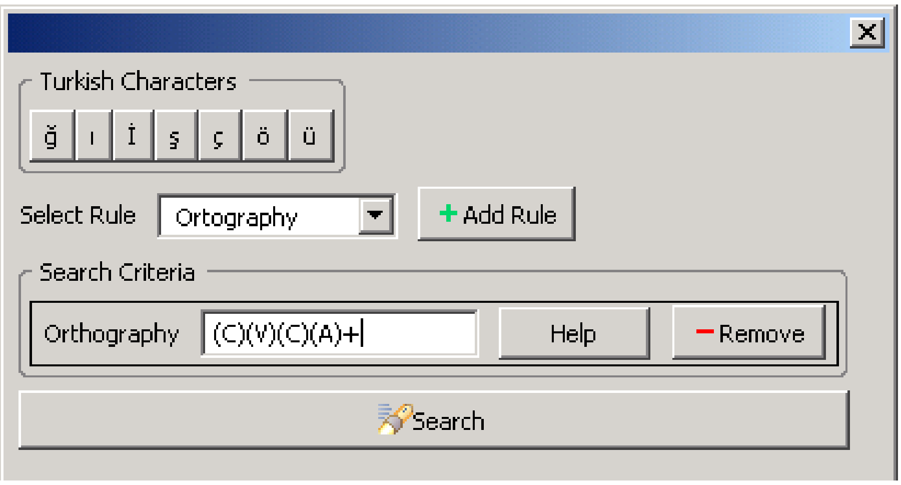
Examples

The following examples will clarify the usage of this rule and Figure 3.14 presents the view in search panel.

- To locate words that contain `lik` anywhere in the word, the following expression can be used: `(A)*lik(A)*`
- To locate words that contain `lik` at the end of the word, the following expression can be used. `(A)*lik`
- To locate words that contain `lik` or `rik` anywhere in the word, the following expression can be used: `(A)*(l | r)ik(A)*`
- To locate words that contain a sequence of *vowel+consonant+vowel* at the beginning of the word and have at least four characters, the following expression can be used: `(V)(C)(V)(A)+`

Table 3.2: Basics Of Regular Expressions

Regular Expression	Meaning
`(A)*`	Any character sequence containing the empty sequence
`(A)+`	Any character sequence containing at least one character
`(C)*`	Any consonant sequence containing the empty sequence
`(C)+`	Any consonant sequence containing at least one character
`(V)*`	Any vowel sequence containing the empty sequence
`(V)+`	Any vowel sequence containing at least one character



Select Rule: Ortography

Orthography: (C)(V)(C)(A)+

Figure 3.14: Orthography Rule

3.2.4 Pronunciation Rules

This is the part where one can form rules to find the words with various segmental properties in a Turkish text. For example, one can construct rules to search for the words that do not follow Turkish vowel harmony or to search for the words that have stress before the last syllable. LingBrowser provides three different interfaces to create rules.

The first one uses regular expressions with SAMPA characters along with meta-characters. Some meta-characters that can be used in this rule are listed in Table 3.3 with their corresponding phoneme group.

Usage Examples

1. To search for words that have a syllable *51* (representing `-lı-` in `ballı`) , one can use the expression `-51-`

Table 3.3: Pronunciation Meta-Characters

Meta Character	Description	SAMPA Symbol List
(Z)	Any vowel or consonant	
(C)	Any consonant	
(V)	Any vowel	
(E)	Front Vowels	`i,e,y,2`
(A)	Back Vowels	`1,a,o,u`
(I)	High Vowels	`u,i,y,1`
(R)	Round Vowels	`o,u,y,2`
(S)	Sonarants	`h,l,5,m,n,r,y`
(O)	Obstruents	`b,c,d,f,g,G,Z,k,p,s, S,t,v,w,z,tS,dZ,gj`
(G)	Voiced Stops	`b,c,d,G,dZ`
(K)	Voiceless Stops	`p,t,k,tS`
(B)	Labial Consonants	`p,v,w,b,f,m`
(T)	Non-Labial Consonants	`c,d,g,G,h,Z,k,l,5,n, r,s,S,t,y,z,tS,dZ,gj`

2. To search for words that have a syllable *5i* at the end of the word, one can use the expression `-51$`

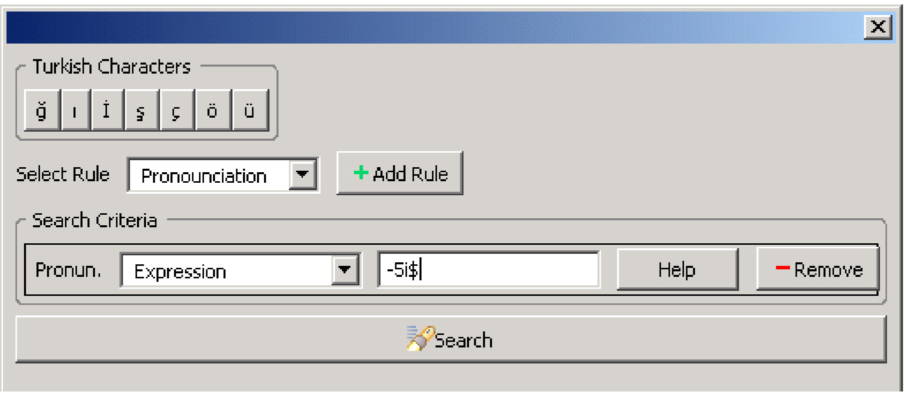
3. To search for words that do not follow vowel harmony, one needs to define two rules one with expression `(E)` (front vowels) and the other one with expression `(A)` (back vowels), thus, LingBrowser will locate words that have both front and back vowels.

4. To search for words that have a syllable with sequence of consonant-vowel-consonant at the beginning of the word, one uses the expression `^(C)(V)(C)-`.

An example rule constructed in search panel is presented in Figure 3.15.

The second type of rule is the same of first type except the user can use surface symbols instead of SAMPA characters to define the rule. The first usage example above searches the words with syllable `-51-` which is in SAMPA format. In this part, user will use the expression `-lı-` where surface character `l` is used instead of SAMPA character `5` and `ı` is used instead of `1`. The details of relation between pronunciation characters and surface characters is detailed in [4].

The third type of rule is for searching words according to the stress location in the pronunciation. One will choose the stress mark location which can be (at last syllable) or (before last syllable) from a drop-down box to define the rule.



Select Rule: Pronounciation

Pronun. Expression -5i$

Figure 3.15: Pronunciation Rule

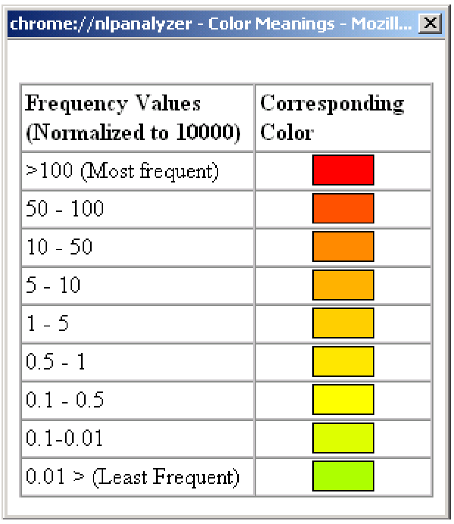
3.3 Additional Features

3.3.1 Word Coloring

LingBrowser has the ability to colorize the words in a loaded text according to their general usage frequency in Turkish. There are nine predefined frequency range groups and each group has its own color. If one uses the ”Colorize Text” option provided by LingBrowser, the words will be colorized according to their groups. When one hovers a colorized word with mouse, a tool tip will be shown presenting the frequency information about the word. Also, one may access the meaning of colors through a color information window (Figure 3.16). Note that the frequency information of some words may not be present, thus they will not be colorized by this functionality. After the coloring process, source document will look like Figure 3.17

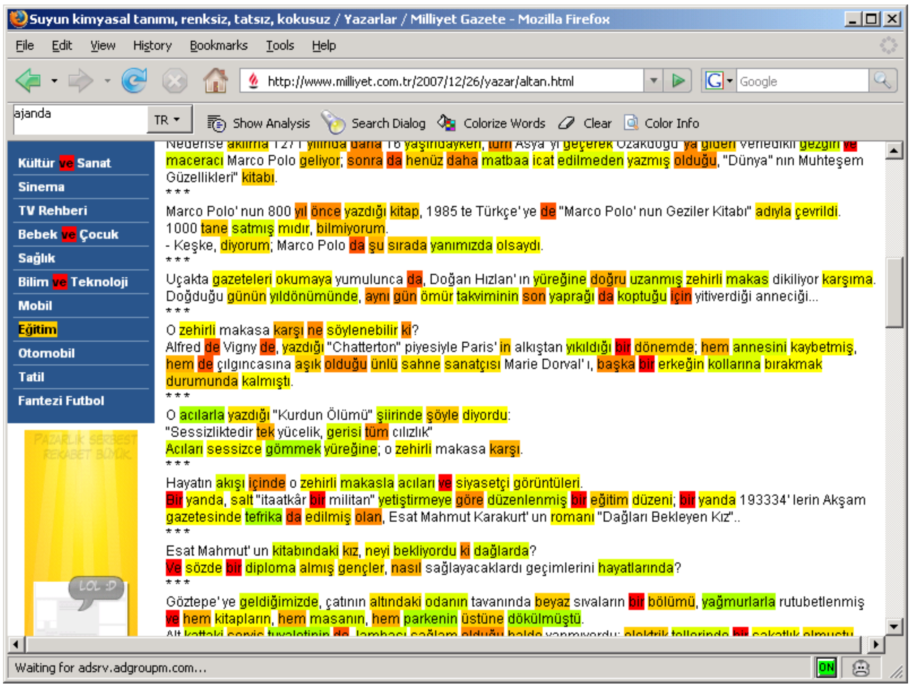
3.3.2 Search in TELL Database

Turkish Electronic Living Lexicon (TELL) was developed in the linguistics department University of California Berkeley by a team leaded by Sharon Inkelas. TELL is a database of 30000 root words and the words constructed from these root words with additions of various suffixes. When a user activates the ”Search in TELL” button in tool-bar of LingBrowser, a search panel that performs a search in TELL database will be opened. This is the same search panel as Section 3.2, but it performs the search operations in the TELL Database.


chrome://nlpanalyzer - Color Meanings - Mozill...

Frequency Values (Normalized to 10000)	Corresponding Color
>100 (Most frequent)	🟥
50 - 100	🟧
10 - 50	🟧
5 - 10	🟧
1 - 5	🟨
0.5 - 1	🟨
0.1 - 0.5	🟨
0.1-0.01	🟩
0.01 > (Least Frequent)	🟩

Figure 3.16: Meanings Of Colors



Nedense aklıma 1271 yılında daha 16 yaşındayken, tüm Asya'yı geçerek Ortadoğu'ya giden Venedikli gezgin ve maceracı Marco Polo geliyor; sonra da henüz daha matbaa icat edilmeden yazmış olduğu, "Dünya" nın Muhteşem Güzellikleri" kitabı.
* * *
Marco Polo' nun 800 yıl önce yazdığı kitap, 1985 te Türkçe' ye de "Marco Polo' nun Geziler Kitabı" adıyla çevrildi.
1000 tane satmış mıdır, bilmiyorum.
- Keşke, diyorum; Marco Polo da şu sırada yanımızda olsaydı.
* * *
Uçakta gazeteleri okumaya yumulunca da, Doğan Hızlan' ın yüreğine doğru uzanmış zehirli makas dikiliyor karşıma.
Doğduğu günün yıldönümünde, aynı gün ömür takviminin son yaprağı da koptuğu için yitiverdiği anneciği...
* * *
O zehirli makasa karşı ne söylenebilir ki?
Alfred de Vigny de, yazdığı "Chatterton" piyesiyle Paris' in alkıştan yıkıldığı bir dönemde; hem annesini kaybetmiş, hem de çılgıncasına aşık olduğu ünlü sahne sanatçısı Marie Dorval' ı, başka bir erkeğin kollarına bırakmak durumunda kalmıştı.
* * *
O acılarla yazdığı "Kurdun Ölümü" şiirinde şöyle diyordu:
"Sessizliktedir tek yücelik, gerisi tüm cılızlık"
Acıları sessizce gömmek yüreğine; o zehirli makasa karşı.
* * *
Hayatın akışı içinde o zehirli makasla acıları ve siyasetçi görüntüleri.
Bir yanda, salt "itaatkâr bir militan" yetiştirmeye göre düzenlenmiş bir eğitim düzeni; bir yanda 193334' lerin Akşam gazetesinde tefrika da edilmiş olan, Esat Mahmut Karakurt' un romanı "Dağları Bekleyen Kız"..
* * *
Esat Mahmut' un kitabındaki kız, neyi bekliyordu ki dağlarda?
Ve sözde bir diploma almış gençler, nasıl sağlayacaklardı geçimlerini hayatlarında?
* * *
Göztepe' ye geldiğimizde, çatının altındaki odanın tavanında beyaz sıvaların bir bölümü, yağmurlarla rutubetlenmiş ve hem kitapların, hem masanın, hem parkenin üstüne dökülmüştü.
Alt kattaki servis tuvaletinin de lambası sağlam olduğu halde yanmıyordu; elektrik tellerinde bir sakatlık olmuştu

Figure 3.17: Word Coloring

Chapter 4
IMPLEMENTATION OF LINGBROWSER

4.1 The Software Architecture

Our first concern in implementation is that the resulting software should utilize HTML context and it should be platform-independent. The main source of HTML is obviously the Internet, so our software should also possess web browsing functionalities for general Internet usage. Other than browsing the Internet, one may also need other features like bookmarking, security preventions, etc. Currently there are many HTML rendering toolkits available, but adding other browsing functionalities to these toolkit will be very time consuming and will be like reinventing the wheel. As a result, instead of implementing a basic web browser that only renders HTML pages, we decided to implement LingBrowser as an add-on for the powerful, pervasive, cross-platform web browser, Mozilla-Firefox[12].

However, this decision have some drawbacks. We are limited with the programming languages that we are going to use because an add-on for Mozilla-Firefox can be implemented only by using Javascript along with XUL (XML User Interface Language)[17]. Javascript was originally designed to create dynamic web pages by manipulating the source of the web page. Thus, it does not have interfaces like database connectivity which we need to use heavily during the implementation. Also, XUL is just for designing graphical user interfaces. On the other hand, Javascript recently gained the ability to exchange XML and HTML data with external resources with emerging of Ajax technology (Asynchronous Javascript and XML) [16]. Ajax is able to send HTTP (Hyper Text Transfer Protocol) requests to a web server and handle the answers for those requests.

At this point, we are able to extract raw data with Javascript from the source HTML text, send the raw data to a server, gather the processed data with help of Ajax framework and we can implement user interfaces to present the processed data in XUL. Remaining part in implementation is a web server that will handle the requests from the browser add-on application.

We implemeted the server side application in Java programming language using the Sun's Java Servlet Technology [14] and the server side runs on Apache/Tomcat application server [15]. Since, Java is a cross-platform programming language, server side can run on any hardware and operating system platforms. On the server, we used MySQL [13] as the database application which can run on different platforms.

To summarize, LingBrowser is the composition of a client application which is a part of web browser and a server application.

4.2 Software Design

We can divide the software design of LingBrowser into three main components: Client-Server Communication Interface, Client Side and Server Side.

4.2.1 Client-Server Communication Interface

In this part, we will introduce two controllers: one is on server side and the other one is on client side. These controllers construct the communication between the server and the client. Furthermore they work as a decision maker. The communication between the server and the client is asynchronous since we are using Ajax framework. So, the main job of these controllers is to channel the requests and results of the requests to the appropriate components of the software.

The Client Controller

Client controller is *XMLHttpRequest* object of Ajax framework. It opens a HTTP (Hyper Text Transfer Protocol) channel to server, posts the parameter and waits for the answer from the server. When the answer, which is either in HTML(Hyper Text Markup Language) or XML (Extended Markup Language), arrives from the server, it channels the message to appropriate handler.

We need to define at least two javascript functions to activate the controller: a function for posting the data and a function for handling the response. Below is a simplest javascript snippet that shows the usage of Client Controller.

```
var xmlHttp;
var resultXML;

function activateClientController(params)
{
        var url = ”http://”+server+”/NLPAnalyzer/nlpajax”; //URL of the server controller

        var postRequest = ”paramName=”+params; // Any data that we want to send to server.

        xmlHttp=new XMLHttpRequest(); // create a Client Controller Object

        xmlHttp.open(”POST”, url , true); //Opens the HTTP channel

        //Set the encoding for data
        xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

        // set the name of the handler method which will handle the response
        xmlHttp.onreadystatechange = handler;

        xmlHttp.send(postRequest); //sends the data

}

function handler()
{

        if(xmlHttp.readyState==4) //check if the answer arrives
        {
        // check the error status. 200 is the HTTP status code for success
                if (xmlHttp.status == 200) {


                resultXML = xmlHttp.responseXML;
                // we can use xmlHttp.responseHTML if we are waiting an HTML response

                ...
                ... Process the resultXML variable for needed functionality
                ...

     }
     else
     {
        //if there is an error during HTTP Call handle it here
                alert(”Html Error.. ”+xmlHttp.status);
     }

    }

}
```

The Server Controller

The server controller is a simple servlet class of Java named *web.NLPAjax*. This class waits for the HTTP POST request from the client and extracts the parameters from the request and channel the parameter to the corresponding processor unit.

Adding Functionality

After the implementation communication interface, it is easy to plug-in additional functionality on this communication. We just need to implement client side components that process XML or HTML files and server side components that generates XML or HTML responses.

4.2.2 Server Side Components

On the server, we have two layers other than the controller which are the annotation layer and the database layer.

An annotator takes a string as input and outputs an XML or HTML text. On the annotation layer, there are four types of annotators that we implemented:

1. Annotator for single word exploration
2. Annotator for search engine
3. Annotator for frequency coloring
4. Annotator for search in the TELL Database

With simple changes on Server Controller, other functional units can be added to this layer.

On the database layer, there are three types of databases that work with annotators: the WordNet database for translation of root word, frequency database for words and transducer based databases for linguistic analysis. In Section 4.3, we present the details of databases.

4.2.3 Client Side Components

On the client side, we have four functional components which are plugged to the client controller. These components are listed as *Single Word Exploration Window*, *Search Panel*, *Word Colorer* and main tool bar for activating these components.

The client side components have two subcomponents; graphical user interface and logical units. The initial user interface is defined by XUL (XML User interface language). An XUL file defines the locations of buttons, menu items, input boxes and determines listener methods for events like pressing a button.

Logical units of a client side components are the Javascript methods which implement all functionality by modifying user interfaces, extracting the data from source document, communicating with Client Controller and manipulating the source text according to the results from Client Controller.

4.2.4 The Execution Flow

Execution flow of LingBrowser can be summarized in following steps:

1. Client side components extract the raw data from the source or inputs and send the raw data to the client controller.

2. Client controller sends the data and the appropriate analysis type to server controller.

3. When the data arrives to the server controller, the server controller decides which annotator will be used according to the analysis type sent by client controller and sends the data to appropriate annotator.

4. The annotator makes the analysis by employing queries to the databases, places the results of analysis in an HTML or XML document and sends this processed data to the server controller.

5. Server controller sends the processed data to the client controller.

6. Client controller forward the resulting data to corresponding client side component.

7. When the client side component gets the data, it processes data if it is needed (for instance search panel performs a search) and presents last form of the data to the user

Figure 4.1 represents this execution flow.

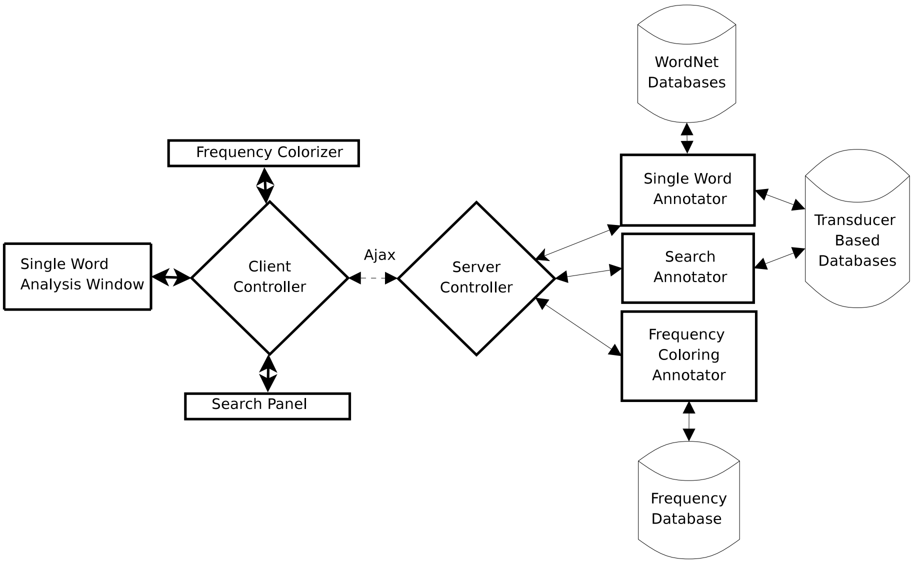


Figure 4.1: LingBrowser data flow diagram

4.3 Populating Databases

LingBrowser makes use of three types of databases. Finite State Transducer Based Database, Translation database and Frequency Database.

4.3.1 Transducer Based Database

Although, we have extensively used finite state transducer technology, we used it in an indirect way. Currently we do not have a finite state transducer runtime library that will do an on the fly analysis, thus we populated a database of analysis from a large set of words that is integrated to the server side to achieve a proof of concept software. When a runtime library is available, this process can be easily altered that we can replace this database with the runtime library. We used Xerox FST tool to construct the transducers and all the transducers that are used to populate these databases are derived from a core morphological analyzer [1]. We used this core morphological analyzer with additional finite state transducers to extract all the relevant representations such as surface morpheme structure, pronunciation representation, etc.

We have four database tables in this database: Feature-Lexical table, Surface Morpheme Structure table, Lexical-Surface Alignment table, Feature-Pronunciation table. We used simple table structures that tables have two-columns: word and analysis. Sample table example for Feature-Lexical is presented on table 4.1. Sample analysis for other tables will be detailed in section 4.4.

Table 4.1: Feature-Lexical Database Table

Word	**Analysis**
kitabına	(kitab)kitap+Noun+A3sg(+Hn)+P2sg(+yA)+Dat
kitabına	(kitab)kitap+Noun+A3sg(+sH)+P3sg(+nA)+Dat

4.3.2 Word Translation Database

To populate the translation database we used English WordNet [2] glossaries and Turkish WordNet [3]. One of the objectives of English WordNet project was assigning a interlingual index number for each word. Turkish WordNet aligns with the English WordNet, thus we find the interlingual index number of the words from the Turkish WordNet and find translation of the word in English WordNet using this interlingual index number. Note that a word can have multiple interlingual index numbers that corresponds to different interpretations, so we store all the possible translations along with the part of speech tagging of the word. As a result our database table consists of three columns: word itself, part-of-speech tagging and the translation.

Table 4.2: Word Translation Database Table

Word	**Part-of-Speech**	**Translation**
yaz	Noun	Summer season
yaz	Verb	To write

4.3.3 Frequency Database

From a very large corpus of Turkish words, we have computed the frequency of each unique word and stored these frequencies in a database. Our first concern is not the calculation of the exact frequencies, but to find most common words. So, the frequency of a word in a large corpus will give us a general idea about how common a word is. We store the calculated frequencies in a table with two columns where the columns are word and frequency value which is normalized to 10,000.

4.4 Implementation of Single Word Exploration

Once a word is selected from the browser or entered as input at the tool-bar of LingBrowser add-on, the selected word is sent directly to the Controller on server side with a parameter that indicates that a single word exploration will be performed. Then, controller sends the input word to the *Single Word Annotator (SWA)*. Single Word Annotator performs the annotations via series of processes:

- Morphology-Lexical Morpheme Alignment Annotation
- Morphology Annotation
- Lexical Morpheme Structure Annotation
- Surface Morpheme Structure Annotation
- Lexical-Surface Morpheme Alignment Annotation
- Morphology-Pronunciation Annotation
- Pronunciation Annotation

- Translation Annotation

After these processes, SWA places all the annotated results into an HTML template and sends the resulting HTML document to the *Controller*. HTML gives us the ability to present the data interactively, such as showing-hiding information, pop-up windows, playing sounds, etc.

4.4.1 Morphology-Lexical Morpheme Alignment Annotation

First, SWA queries the input word in Feature-Lexical Database. The results of the query is annotated with HTML tags to present the results in an HTML table . Also, SWA employs some additional HTML tagging to be able to hide morphology or lexical parts of the results by simple split and replace operations performed with regular expressions. For instance, database query results for word *kitabına* is :

(kitab)`kitap+Noun+A3sg`(+Hn)`+P2sg`(+yA)`+Dat`
(kitab)`kitap+Noun+A3sg`(+sH)`+P3sg`(+nA)`+Dat`

The annotated result will be:

```
<table>
<tr>
        <td><b>1)</b></td>
        <td>
        <span class='onlylexical '  style='display:none'>kitab</span>
  <span class='lexical'>(kitab)</span>
        <span class='feature'>kitap+Noun+A3sg</span>
        <span class='onlylexical 'style='display:none'>+Hn</span>
        <span class='lexical'>(+Hn)</span>
        <span class='feature'>+P2sg</span>
        <span class='onlylexical 'style='display:none'>+yA</span>
        <span class='lexical'>(+yA)</span>
        <span class='feature'>+Dat</span>
        </td>
</tr>
<tr >
        <td><b>2)</b></td>
        <td>
        <span class='onlylexical '  style='display:none'>kitab</span>
        <span class='lexical'>(kitab)</span>
        <span class='feature'>kitap+Noun+A3sg</span>
        <span class='onlylexical 'style='display:none'>+sH</span>
        <span class='lexical'>(+sH)</span><span class='feature'>+P3sg</span>
        <span class='onlylexical 'style='display:none'>+nA</span>
        <span class='lexical'>(+nA)</span>
        <span class='feature'>+Dat</span>
        </td>
</tr>
</table>
```

Note that this HTML code is given to show how annotation process results. In other parts, HTML annotations will not be detailed.

4.4.2 Morphological Analysis

For annotating morphological analysis of a word, the query results from 4.4.1 will be used. We can derive the morphological analysis of the word by simply removing the parts in parenthesis from the query results. For the example in 4.4.1, if we remove the parts in parenthesis, we will have the following morphological analysis:

```
kitap+Noun+A3sg+P2sg+Dat
kitap+Noun+A3sg+P3sg+Dat
```

After extracting the morphological analysis, we may perform the HTML annotation to form the presentation of morphological analysis. In this section, annotation includes placing HTML tags for showing and hiding pop-up windows for explanations of morphological features.

4.4.3 Lexical Morpheme Structure

In this section, we also use the database query results from 4.4.1. In contrast to 4.4.2, we will extract the parts in parenthesis and join them to form the Lexical Morpheme Structure of the input word. During this operation, descriptions of morphemes are also gathered. For the example in 4.4.1, we will acquire the resulting analysis as:

```
kitab+Hn+yA
kitab+sH+nA
```

Each morpheme in this representation will be placed between HTML tags, so that the meanings of morpheme can be shown as a pop-up window.

4.4.4 Surface Morpheme Structure

The annotation of Surface Morpheme Structure is straight forward. SWA looks up the words in Surface Database and simply places the results in an HTML Table.

4.4.5 Lexical-Surface Morpheme Alignment

We query the input word in Lexical-Surface Alignment table for this section and do simple HTML formatting along with text processing. As an example, for word `kedisine`, query result will be:

```
k*ke*ed*di*i+*0s*sH*i+*0n*nA*e
```

Then we extract the feasible pairs as (k,k), (e,e), (d,d), (i,i), (+,0), (s,s), (H,i), (+,0), (n,n), (A,e). After, we determine the corresponding explanations that are going to be presented in pop-up window. For instance pair (H,i) matches with the explanation *"Lexical H realized as surface i, since the last surface vowel is one of e,i. "*. (See Appendix C for a complete list). Then we do HTML formating and return the result.

4.4.6 Morphology-Pronunciation Alignment

This part is same as 4.4.1 except SWA looks up the word in Morphology-Pronunciation database. A similar annotation process is performed subsequently to hide Morphology or Pronunciation parts. As an example, query result for word *ajanda* is:

```
(a - "Z a n - d a )ajanda+Noun+A3sg+Pnon+Nom
(a - Z a n )ajan+Noun+A3sg+Pnon(- "d a )+Loc
```

4.4.7 Pronunciation

To get the possible pronunciations of a word, SWA uses the query results from 4.4.6. As in the example of the previous section, parts in parentheses correspond to the pronunciation of the words. These parts are extracted and combined to form the pronunciation. In the annotation step, each character placed in HTML tags so that a pop-up window with related information about character can be shown when mouse hovers on them. Another functionality of LingBrowser is the ability to display pronunciation in IPA. We get the query results in SAMPA from the database. There is a one-to-one mapping between SAMPA and IPA, thus we simply change all SAMPA characters to IPA characters to convert SAMPA pronunciation to IPA pronunciation. When we get the IPA form of the pronunciation, we simply perform the same annotations to construct the result.

4.4.8 Translation of root words via WordNet

To have the translation of root, first we must extract the root of the word. We already have the morphological analysis of the word from 4.4.2, thus getting the root of the word is straightforward. Then, SWA looks up the root of the word in WordNet database. The results for the query include the translation and part-of-speech tag of the root. At the end, these results are converted into HTML form and returned to SWA.

4.5 Implementation of The Search Engine

Search Engine is one of the core components of LingBrowser that it enables the users of LingBrowser to locate examples for linguistic rules of Turkish language in a Turkish text. To perform a search, LingBrowser analyzes all the words in a text, stores the resulting analysis and performs the search action on these analysis. This will be realized in four steps:

1. LingBrowser processes the text in web browser to sort out the words which will be used in search process.

2. The words gathered in preprocessing phase will be sent to server side to be analyzed.

3. The Server side analyze the words and send the processed data back to client.

4. Analysis of the words are stored in client side and then LingBrowser will perform the search over these analysis by utilizing regular expressions which are constructed according to the inputs of users.

Basic data flow of this process summarized in Figure 4.2 and the process will be detailed in following subsections.

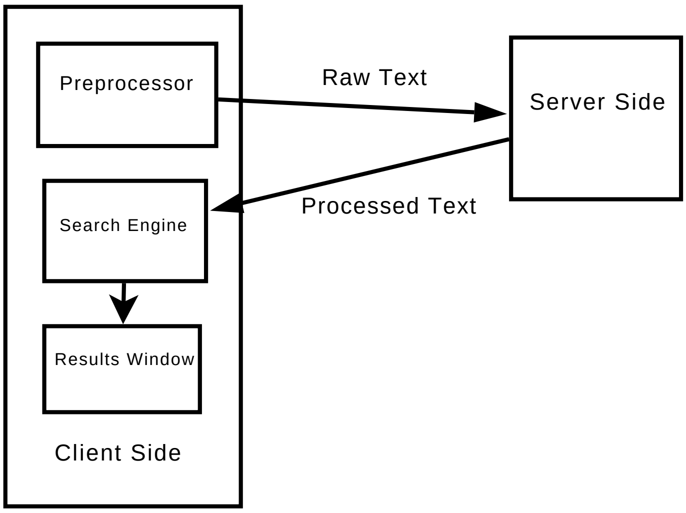


Figure 4.2: LingBrowser data flow diagram for search

4.5.1 Text Preprocessing

Since, the main source of text that will be utilized by LingBrowser is web pages in HTML. In an HTML page, there are HTML tags, HTML comments, various scripts and the main text of page. A very simple HTML page will look like as the following:

```
<html>

<head>
 <title>This is the title of page</title>
</head>
<body>
<!-- This is a comment -->
 <div>Merhaba, bu sayfada haberler bulunur.</div>
 <script>
  function foo()
  {
   alert(); // basic javascript
  }
 </script>

</body>

</html>
```

For the above example, we will get rid of all the HTML components by straightforward text processing and obtain the raw text as

`Merhaba, bu sayfada haberler bulunur.`

At this point, we remove the punctuation marks and the resulting text will be:

`Merhaba bu sayfada haberler bulunur`

Then, this result is sent to *Server Side*.

4.5.2 Linguistic Analysis

In this phase, raw text is tokenized. Then for each word, mophological analysis, pronunciation, lexical morpheme structure , surface morpheme structure are obtained with similar operations that are described in section 4.4. After this point, the results are annotated in XML. Annotating the results in XML enables us to use XML parsing and editing utilities of underlying web browser Mozilla-Firefox. For the example in Section 4.5.1, the annotated document will look like as the following:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<root>
 <word value=”Merhaba”>
   <feature>
     <featureitem>merhaba+Noun+A3sg+Pnon+Nom</featureitem>
   </feature>
   <lexical>
     <lexicalitem>merhaba</lexicalitem>
   </lexical>
   <pronoun>
     <pronounitem>”m e r - h a - b a</pronounitem>
   </pronoun>
   <surface>
     <surfaceitem>merhaba</surfaceitem>
   </surface>
 </word>
 <word value=”bu”>
   <feature>
     <featureitem>bu+Pron+Demons+A3sg+Pnon+Nom</featureitem>
     <featureitem>bu+Det</featureitem>
   </feature>
   <lexical>
     <lexicalitem>bu</lexicalitem>
   </lexical>
   <pronoun>
     <pronounitem>”b u</pronounitem>
   </pronoun>
   <surface>
     <surfaceitem>bu</surfaceitem>
   </surface>
 </word>
 <word value=”sayfada”>
   <feature>
     <featureitem>sayfa+Noun+A3sg+Pnon+Loc</featureitem>
   </feature>
   <lexical>
     <lexicalitem>sayfa+dA</lexicalitem>
   </lexical>
   <pronoun>
     <pronounitem>s a j - f a - ”d a</pronounitem>
   </pronoun>
   <surface>
     <surfaceitem>sayfa+da</surfaceitem>
   </surface>
 </word>
 <word value=”haberler”>
   <feature>
     <featureitem>haber+Noun+A3pl+Pnon+Nom</featureitem>
   </feature>
   <lexical>
     <lexicalitem>haber+lAr</lexicalitem>
   </lexical>
   <pronoun>
     <pronounitem>h a - b e r - ”l e r</pronounitem>
   </pronoun>
   <surface />
 </word>
 <word value=”bulunur”>
   <feature>
```

```
    <featureitem>bul+Verb^DB+Verb+Pass+Pos+Aor^DB+Adj+Zero</featureitem>
    <featureitem>bul+Verb^DB+Verb+Pass+Pos+Aor+A3sg</featureitem>
    <featureitem>bulun+Verb+Pos+Aor^DB+Adj+Zero</featureitem>
    <featureitem>bulun+Verb+Pos+Aor+A3sg</featureitem>
  </feature>
  <lexical>
    <lexicalitem>bul+Hn+Hr</lexicalitem>
    <lexicalitem>bulun+Hr</lexicalitem>
  </lexical>
  <pronoun>
    <pronounitem>b u - 5 u - "n u r</pronounitem>
  </pronoun>
  <surface>
    <surfaceitem>bul+un+ur</surfaceitem>
    <surfaceitem>bulunur</surfaceitem>
  </surface>
 </word>
</root>
```

As one can see, results for each word is grouped in `<words>` tags and analysis of the word is structured in these tags accordingly.

4.5.3 The Search Process

At this point, we have a processed text that a search can be performed on. The search process begins with constructing regular expressions from the rules defined by the user. Sample regular expressions for each type of rule are presented in Table 4.3.

Table 4.3: Sample Regular Expressions

Type	Input	Regular Expression
Morphology (root POS)	Noun	$\hat{}[\hat{}\backslash+]+Noun\backslash+?$
Morphology (root)	gel	$\hat{}gel\backslash+$
Morphology (hasFeature)	PastPart	$\hat{}\backslash S+\backslash+PastPart(\backslash+)?$
Pronunciation (Expression)	-5i-	-5i($|-)
Pronunciation (Ascii Expression)	-li-	-(5|l)i($|-)
Lexical	DA	d

After the construction of a regular expression, we traverse the XML document obtained from Section 4.5.2 and try to match the corresponding analysis. If we can match all the regular expressions with at least one interpretation, than we flag the word as a result and also flag the interpretation to be able to highlight them in the search result window. Then, the flagged words and analysis of those words is sent to result window. Since we know which interpretation matches all the rules that applies, we can simply highlight that interpretation of the selected word in search result window. Note that, we are not able to construct an indexing mechanism on the analysis, since we are using multiple types of rules and nature of these rules do not allow us to apply any indexing methods. However, this situation won't cause any performance problems, since a web page generally does not have more than 1000 words.

4.6 Implementation of Frequency Coloring

The data flow of this section is similar to previous section. We can study the implementation in three phases: Text Processing, Finding Frequencies and Coloring Words.

4.6.1 Text Processing

Implementation described in subsection 4.5.1 will be used in this part.

4.6.2 Finding Frequencies

On the server side, when the raw text arrives, we look up the frequency database for each word in the text. According to the value of frequencies, each word is assigned to a color group and the results are sent to the client in an XML format. Generated XML document will look like the following:

```
<?xml version=”1.0” encoding=”UTF−8”?>
<root>
 <word value=”ve” color=1 />
 <word value=”daha” color=2 />
 <word value=”mutluluklar” color=3 />
</root>
```

4.6.3 Coloring Words

In this phase, first we parse the XML document and construct a Javascript built-in *Array* object which is a hash table implementation. As a result of the hash table property of Javascript *Array* object, we do not need an additional indexing structure to increase the performance. This array object stores the data as key-value pairs. In our case, words will be *key* and color groups will be *value*. After this process, we simply traverse the source document loaded in LingBrowser(HTML document) and go over all the words. We look up the assigned group of each word from the constructed Array object. If the word exists in the Array object, then we replace the word with an HTML fragment which will color the word. A Generic HTML Fragment for this case will be:

```
<span class="clrzd"
  title="[%frequencyinfo%]"
  style="background-color:[%bgcolor%]; color: black;">[%word%]</span>
```

We generate the HTML fragment of a word by replacing [%frequencyinfo%], [%bgcolor%] and [%word%] with the corresponding information.

4.7 Implementation of Search in TELL Database

Tell database has nearly 85.000 words. First we analyze all these words and create a huge XML File (30MB) in the format that is described in 4.5.2 on the server side. When the tell search is activated on server side, this file is downloaded to the client side only once and stored at the client side. When the search process is activated again, we use the previously downloaded data. Previously explained search operations can now be performed on this file. Since, we designed the search operations for 1000-2000 words, this functionality have some performance problems. A complex search can take 5 minutes or more since we have to check for all 85.000 words and their analysis. We are trying to match regular expressions while performing the search and there is no applicable indexing structures for this type of operations in Javascript. However, in near future, we are planning to move the search operations for this functionality to the server side where we can use the indexing features of MySQL database engine.

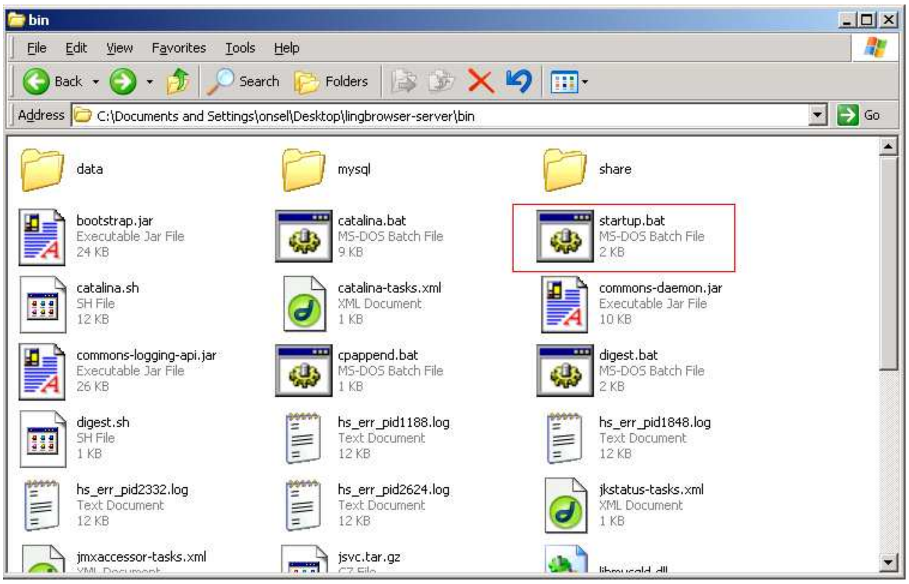
Chapter 5
A SESSION WITH LINGBROWSER

In this chapter, we will present a guide to start a session with LingBrowser in a Microsoft Windows environment to demonstrate the capabilities of our system. This guide can be applied to other environments with minor changes.

5.1 Installation and Configuration

5.1.1 Server Side

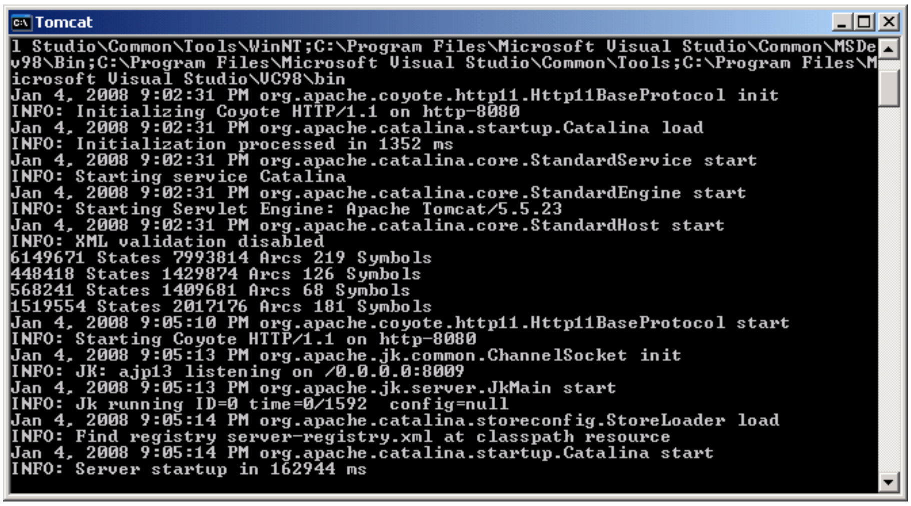
We pre-configured the server side and packed the LingBrowser application, the MySQL database, the java runtime environment and the tomcat application server into a single zip file. To start the server, one just needs to unzip this zipped file and run the "startup.bat" file in *bin* folder (See Figure 5.1).



Address C:\Documents and Settings\onsel\Desktop\lingbrowser-server\bin

Figure 5.1: Run Server

This will start Tomcat Server using the Java Runtime Environment that comes with the package and will start the MySQL database server (See Figure 5.2).



```
l Studio\Common\Tools\WinNT;C:\Program Files\Microsoft Visual Studio\Common\MSDe
v98\Bin;C:\Program Files\Microsoft Visual Studio\Common\Tools;C:\Program Files\M
icrosoft Visual Studio\VC98\bin
Jan 4, 2008 9:02:31 PM org.apache.coyote.http11.Http11BaseProtocol init
INFO: Initializing Coyote HTTP/1.1 on http-8080
Jan 4, 2008 9:02:31 PM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 1352 ms
Jan 4, 2008 9:02:31 PM org.apache.catalina.core.StandardService start
INFO: Starting service Catalina
Jan 4, 2008 9:02:31 PM org.apache.catalina.core.StandardEngine start
INFO: Starting Servlet Engine: Apache Tomcat/5.5.23
Jan 4, 2008 9:02:31 PM org.apache.catalina.core.StandardHost start
INFO: XML validation disabled
6149671 States 7993814 Arcs 219 Symbols
448418 States 1429874 Arcs 126 Symbols
568241 States 1409681 Arcs 68 Symbols
1519554 States 2017176 Arcs 181 Symbols
Jan 4, 2008 9:05:10 PM org.apache.coyote.http11.Http11BaseProtocol start
INFO: Starting Coyote HTTP/1.1 on http-8080
Jan 4, 2008 9:05:13 PM org.apache.jk.common.ChannelSocket init
INFO: JK: ajp13 listening on /0.0.0.0:8009
Jan 4, 2008 9:05:13 PM org.apache.jk.server.JkMain start
INFO: Jk running ID=0 time=0/1592  config=null
Jan 4, 2008 9:05:14 PM org.apache.catalina.storeconfig.StoreLoader load
INFO: Find registry server-registry.xml at classpath resource
Jan 4, 2008 9:05:14 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 162944 ms
```

Figure 5.2: Tomcat Window

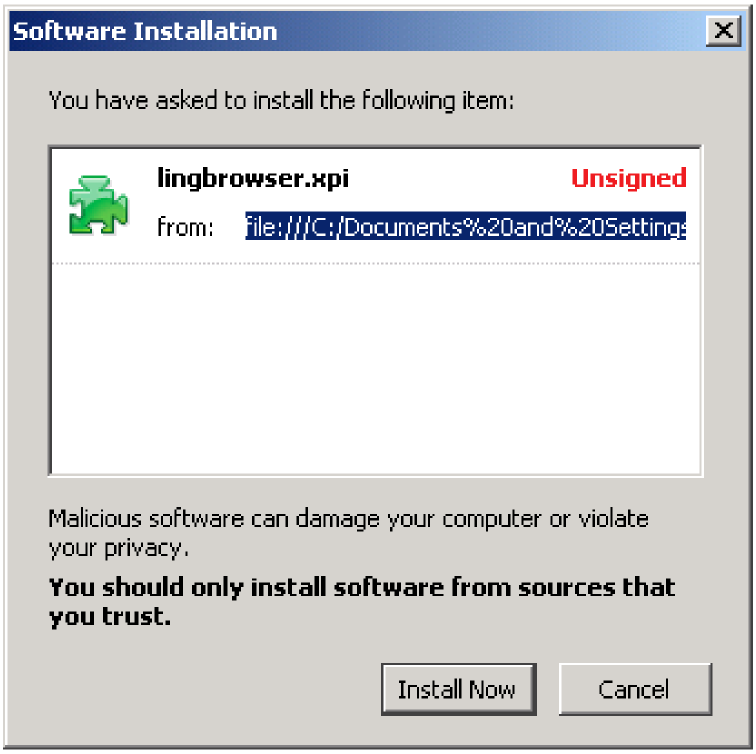
By default Tomcat Application Server uses port 8080, but this can be changed by editing below lines in */lingbrowser-server/conf/server.xml* file.

```
<Connector port="8080" maxHttpHeaderSize="8192"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" redirectPort="8443" acceptCount="100"
connectionTimeout="20000" disableUploadTimeout="true" />
```

To stop the server, one can simply close the Tomcat window.

5.1.2 Client Side

In this part, we assume that the Mozilla-Firefox is installed and running properly on the user's machine. The client side of LingBrowser is packed into a Mozilla Firefox Add-On file whose name is *lingbrowser.xpi*. Curently, our plugin can run on 1.x and 2.0.x versions of Mozilla-Firefox. To install the plugin, one clicks **File→Open File** menu item, selects the lingbrowser.xpi file and clicks *Open* button. This action will open a window which is presented in Figure 5.3. To finish the installation, one clicks install button and restart the Mozilla-Firefox application. After the restart, a toolbar will be added to Mozilla-Firefox. Before using LingBrowser one may need to overview the configuration. We can configure the client side, so that it works with a server which runs on different machine in a configuration window (See Figure 5.4) which can be opened through Add-On configuration panel of Mozilla-Firefox (**Tools→Add-ons** menu item and see also in Figure 5.5). Furthermore, we can configure the size of single word exploration window, we can enable/disable double click functions through this control panel. After the configuration phase, we can



Software Installation

You have asked to install the following item:

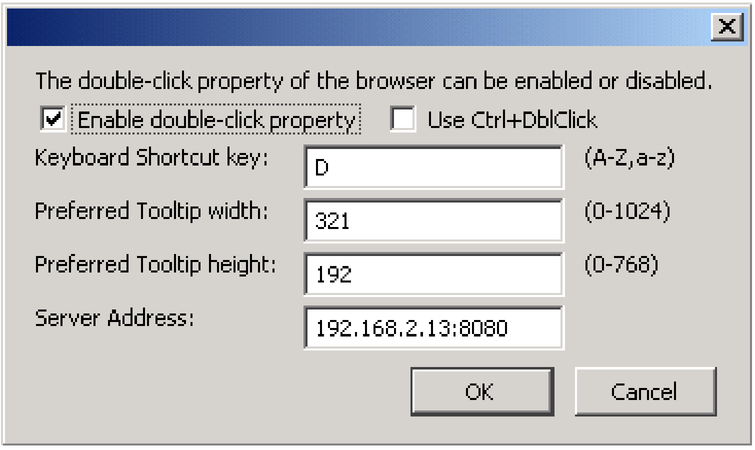
lingbrowser.xpi Unsigned

from: file:///C:/Documents%20and%20Setting

Malicious software can damage your computer or violate your privacy.

You should only install software from sources that you trust.

Figure 5.3: Install Plugin



The double-click property of the browser can be enabled or disabled.

☑ Enable double-click property □ Use Ctrl+DblClick

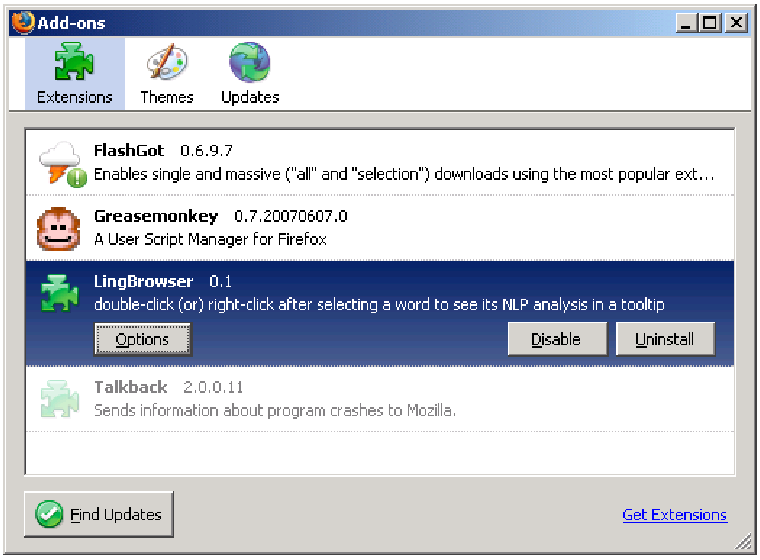
Keyboard Shortcut key: D (A-Z,a-z)

Preferred Tooltip width: 321 (0-1024)

Preferred Tooltip height: 192 (0-768)

Server Address: 192.168.2.13:8080

Figure 5.4: LingBrowser Configuration



FlashGot 0.6.9.7
Enables single and massive ("all" and "selection") downloads using the most popular ext...

Greasemonkey 0.7.20070607.0
A User Script Manager for Firefox

LingBrowser 0.1
double-click (or) right-click after selecting a word to see its NLP analysis in a tooltip

Talkback 2.0.0.11
Sends information about program crashes to Mozilla.

Figure 5.5: Mozilla-Firefox Add-on Configuration Panel

start using LingBrowser. Note that javascript property of the Mozilla-Firefox Application should be enabled.

5.2 Single Word Exploration

One can analyze a word with LingBrowser through three different ways. Firstly, one can double-click the word in browser window. Secondly, one can type the word in the input-box which is located in tool bar (See Figure 5.6) and hit the *Show Analysis* button and as a last one, one can highlight the word and click the "View NLP Analysis" item of right-click menu (See Figure 5.7). Last one can be used to analyse words that cannot be double-clicked like the words that appears in an HTML link.

5.3 Activating Search Process

One can start the search process by clicking the Search Panel button on tool-bar. After the click, user should wait a message indicating the completion of analysis process (See Figure 5.8). By clicking the *OK* button on this message, search panel will be displayed. Analysis process is only executed when a new page is loaded. Once the analysis of a page is done, clicking Search Panel button on the tool bar again will not trigger another analysis process, but will directly open Search Panel.

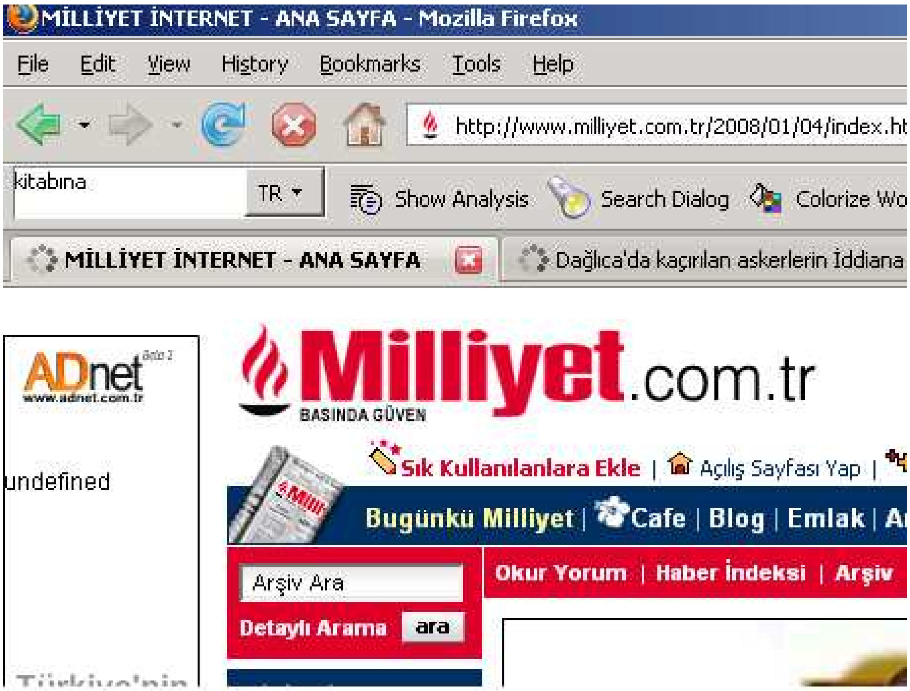

MİLLİYET İNTERNET - ANA SAYFA - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.milliyet.com.tr/2008/01/04/index.ht
kitabına
TR
Show Analysis
Search Dialog
Colorize Wo
MİLLİYET İNTERNET - ANA SAYFA
Dağlıca'da kaçırılan askerlerin İddiana
ADnet
undefined
Milliyet.com.tr
BASINDA GÜVEN
Sık Kullanılanlara Ekle | Açılış Sayfası Yap |
Bugünkü Milliyet | Cafe | Blog | Emlak | A
Arşiv Ara
Detaylı Arama ara
Okur Yorum | Haber İndeksi | Arşiv

Figure 5.6: Input Box

5.3.1 Defining Rules

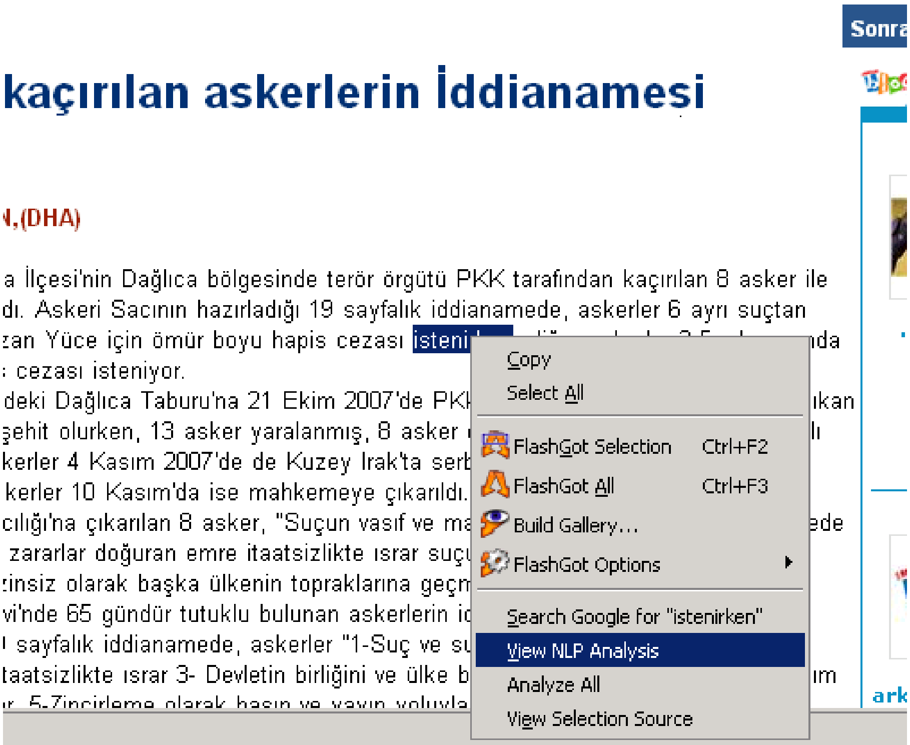
Users can start defining criteria by selecting the type of rule from the drop down box located on search panel and clicking the Add Rule button. Figure 5.9 presents a sample view of the search panel with two rules.

5.3.2 Search Results

After defining the rules, Search button should be clicked and the result of the search will be shown in a new window (See Figure 5.10). On this window, one can now browse the results of the search process.

5.4 Coloring Words

Frequency coloring functionality can be activated by simply clicking Colorize Word button that is on the tool bar of LingBrowser. After colorization of words, one can clear the colors by clicking the Clear button on tool-bar. Another button named **Color Info** will display a information window about the colors and the frequencies that they correspond to upon clicking.

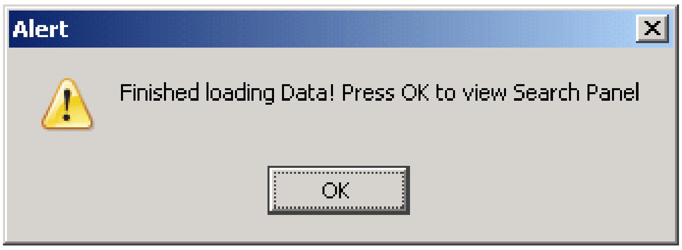


kaçırılan askerlerin İddianamesi

I,(DHA)

a İlçesi'nin Dağlıca bölgesinde terör örgütü PKK tarafından kaçırılan 8 asker ile
dı. Askeri Savcının hazırladığı 19 sayfalık iddianamede, askerler 6 ayrı suçtan
zan Yüce için ömür boyu hapis cezası isteni nda
: cezası isteniyor.
deki Dağlıca Taburu'na 21 Ekim 2007'de PKK ıkan
şehit olurken, 13 asker yaralanmış, 8 asker lı
kerler 4 Kasım 2007'de de Kuzey Irak'ta serb
kerler 10 Kasım'da ise mahkemeye çıkarıldı.
cılığı'na çıkarılan 8 asker, "Suçun vasıf ve ma ede
zararlar doğuran emre itaatsizlikte ısrar suçu
:insiz olarak başka ülkenin topraklarına geçm
vi'nde 65 gündür tutuklu bulunan askerlerin iç
sayfalık iddianamede, askerler "1-Suç ve su
taatsizlikte ısrar 3- Devletin birliğini ve ülke b ım

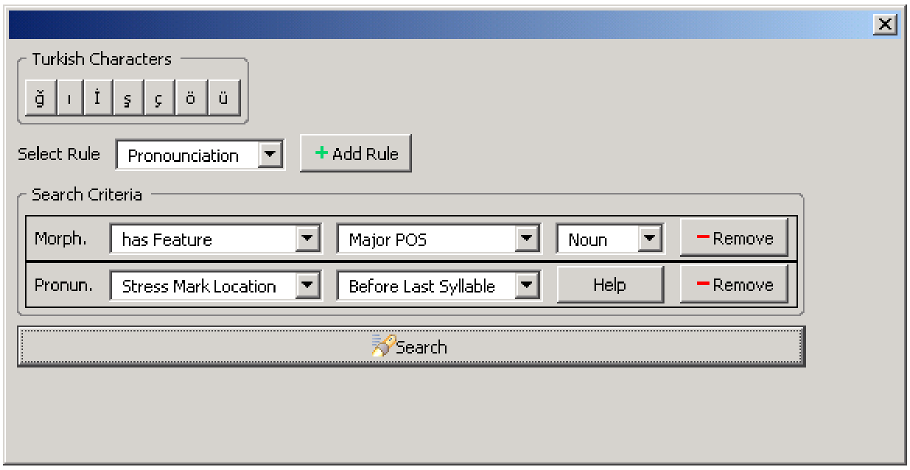
Figure 5.7: Right-Click Menu



Alert

Finished loading Data! Press OK to view Search Panel

Figure 5.8: Analysis Completed Message

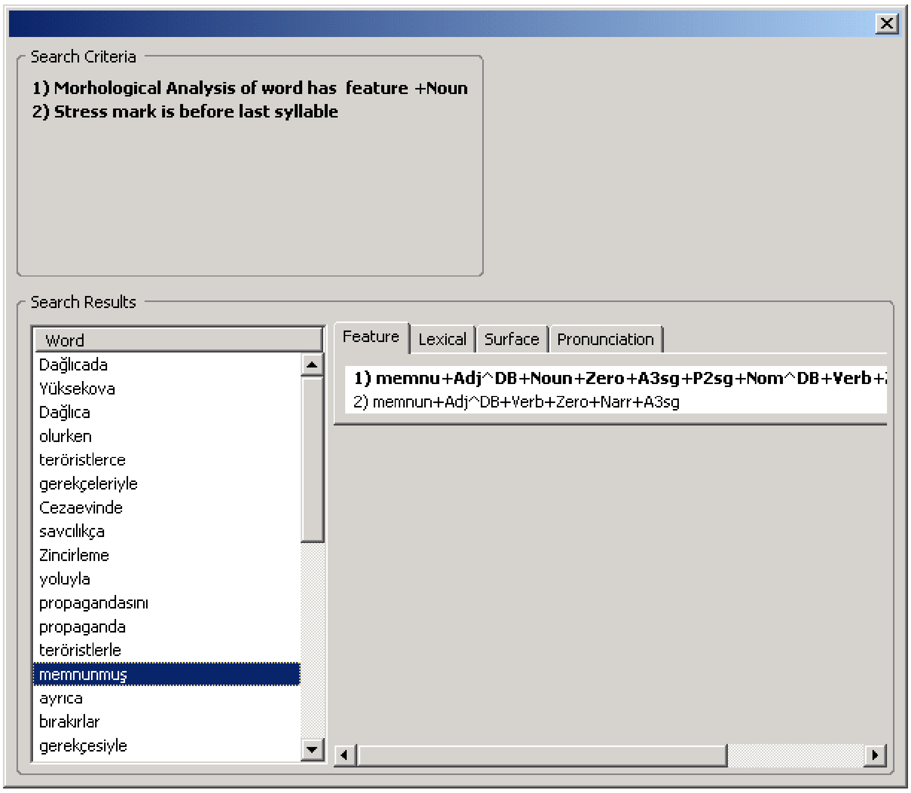


Select Rule: Pronounciation

Morph. has Feature | Major POS | Noun

Pronun. Stress Mark Location | Before Last Syllable

Figure 5.9: Example of Search Rules



1) Morhological Analysis of word has feature +Noun
2) Stress mark is before last syllable

Word
Dağlıcada
Yüksekova
Dağlıca
olurken
teröristlerce
gerekçeleriyle
Cezaevinde
savcılıkça
Zincirleme
yoluyla
propagandasını
propaganda
teröristlerle
memnunmuş
ayrıca
bırakırlar
gerekçesiyle

1) memnu+Adj^DB+Noun+Zero+A3sg+P2sg+Nom^DB+Verb+
2) memnun+Adj^DB+Verb+Zero+Narr+A3sg

Figure 5.10: Search Results.

Chapter 6
CONCLUSIONS

In this thesis, we introduced LingBrowser, an NLP based hypertext browser for linguistic exploration of Turkish words. With LingBrowser, one can reach the morphological analysis and other linguistic properties of a word and locate the words that possess user defined properties in a text.

The important property of LingBrowser is that it can work on HTML pages, thus it can utilize the limitless resources of Internet without relying on previously prepared text. This gives the learners the advantage of choosing the reading materials freely according to their interest areas. Thus, they will be more involved with the text and learning quality will dramatically improve. At this point, let us point out an interesting follow-up work. Imagine a linguist encountering an unknown word while reading a text. He will have two choices. He can immediately look up the structure of the word via LingBrowser, or he can first try to infer the structure from the context and then look up the structure. Obviously, second choice will have a positive effect on learning. One may add a mechanism to LingBrowser that encourages users to first infer the structure from the context. For example, before presenting the results of a query, users may be forced to enter an answer for some properties like morpheme structure and then LingBrowser can show the results in a fashion that users can compare them with their answers.

The functionality of LingBrowser was limited with words in this work. Main reason for that is NLP technology for larger scopes like sentence parser, concordances analyzers, ambiguity resolvers are not mature yet. When these technologies become robust as morphological analyzers are, they can easily be utilized by LingBrowser framework.

Emergence of robust morphological analyzers made NLP to have a place in computer assisted language learning (CALL). However, many CALL systems does not make use of NLP. LingBrowser was developed with the idea that NLP technologies should be used in language learning because classic self-study tools which are based on multiple-choice exercises do not satisfy the advanced learners. We hope that LingBrowser will motivate the linguists to conduct research on Turkish language and greatly increase the number of researches on Turkish.

Chapter 7
APPENDIX A - Turkish Morphological Features

Table 7.1: Major Part-of-Speech

Feature	Explanations
+Noun	Noun
+Adj	Adjective
+Adv	Adverb
+Cond	Condition
+Det	Determiner
+Dup	Onomatopoetic words
+Interj	Interjection
+Ques	Question
+Verb	Verb
+Postp	Postpositive
+Num	Number
+Pron	Pronoun
+Punc	Punctuation

Table 7.2: Minor Part-of-Speech

Feature	Explanations
+Card	Cardinal
+Ord	Ordinal
+Percent	Percentage
+Range	Range
+Real	Real
+Ratio	Ratio
+Distrib	Distribution
+Time	Time
+Inf	Infinitive
+PastPart	Past Participle
+FutPart	Future Participle
+Prop	Proper Noun
+PastPart	Past Participle
+FutPart	Future Participle
+PresPart	Present Particple
+DemonsP	Demonstrative Pronoun
+QuesP	Question Pronoun
+ReflexP	Reflexive Pronoun
+PersP	Personal Pronoun
+QuantP	Quantifying Pronoun

Table 7.3: Nominal Forms

Feature	Explanations
+A1sg	1. singular Person/Number Agreement
+A2sg	2. singular Person/Number Agreement
+A3sg	3. singular Person/Number Agreement
+A1pl	1. plural Person/Number Agreement
+A2pl	2. plural Person/Number Agreement
+A3pl	3. plural Person/Number Agreement
+P1sg	1. singular Possessive Agreement
+P2sg	2. singular Possessive Agreement
+P3sg	3. singular Possessive Agreement
+P1pl	1. plural Possessive Agreement
+P2pl	2. plural Possessive Agreement
+P3pl	3. plural Possessive Agreement
+Pnon	Pronoun (no overt agreement)
+Nom	Nominative
+Acc	Accusative/Objective
+Dat	Dative (to ...)
+Abl	Ablative (from ...)
+Loc	Locative (on/at/in ...)
+Gen	Genitive (of ....)
+Ins	Instrumental (with ...)
+Equ	Equative (by (object) in passive sentences)

Table 7.4: Verb Markers

Feature	Explanations
+Pass	Passive
+Caus	Causative
+Reflex	Reflexive
+Recip	Peciprocal
+Able	able to verb
+Repeat	verb repeatedly
+Hastily	verb hastily
+EverSince	have been verbing ever since
+Almost	almost verbed but did not
+Stay	stayed/frozen while verbing
+Start	start verbing immediately
+Pos	Positive
+Neg	Negative
+Past	Past tense
+Narr	Narrative past tense
+Fut	Future tense
+Aor	Aorist, may indicate, habitual, present, future
+Pres	Present tense
+Desr	Desire/wish
+Cond	Conditional
+Neces	Necessitative, must
+Opt	Optative, let me/him/her verb
+Imp	Imperative
+Prog1	Present continuous, process
+Prog2	Present continuous, state

Table 7.5: Semantic Markers For Derivations

Feature	Explanations
+SinceDoingSo	Since doing so
+As	
+When	
+ByDoingSo	
+While	
+AsIf	
+WithoutHavingDoneSo	
+Ly	corresponds to English slow → slowly
+Since	
+With	
+Without	
+FitFor	
+InBetween	this actually in productive compounding
+Agt	Property of being involved in someway with the stem noun
+Dim	Diminutive
+Ness	as in Red vs Redness
+Become	to become like the noun or adj in the stem
+Acquire	to acquire the noun in the stem

Chapter 8
APPENDIX B - SAMPA CHART FOR TURKISH

Sampa	Orthography	Example		Sampa	Orthography	Example
i	i	kil		e	e	kedi
y	ü	kül		2	ö	göl
u	u	kul		o	o	kol
1	ı	kıl		a	a	kal
p	p	ip		b	b	bal
t	t	ot		gb d	d	dede
c	k	kedi		gj	g	genç
k	k	akıl		g	g	karga
f	f	fare		v	v	ver
s	s	ses		z	z	azık
S	ş	aşık		Z	j	müjde
h	h	hasta		G	ğ	sağır
tS	ç	seçim		dZ	c	cam
m	m	dam		n	n	anı
l	l	lale		5	l	hala
r	r	raf		j	y	yat

Chapter 9
APPENDIX C - SURFACE LEXICAL PAIRS

A:a → Lexical ”A” is realized as a surface ”a” since the last surface vowel is one of ”a,ı,o,u”
A:e → Lexical ”A” is realized as a surface ”e” since the last surface vowel is one of ”e,i,ö,ü”
H:y → Lexical ”H” is realized as a surface ”y” since the last surface vowel is one of ”a,ı”
H:i → Lexical ”H” is realized as a surface ”i” since the last surface vowel is one of ”e,i”
H:u → Lexical ”H” is realized as a surface ”u” since the last surface vowel is one of ”o,u”
H:ü → Lexical ”H” is realized as a surface ”” since the last surface vowel is one of ”ö,ü””
A:0 → Lexical ”A” is deleted on the surface since the next morpheme is the present continuous morpheme ”+Hyor”
a:0 → Surface ”a” is deleted on the surface since the next morpheme is the present continuous morpheme ”+Hyor”
e:0 → Surface ”e” is deleted on the surface since the next morpheme is the present continuous morpheme ”+Hyor”
n:0 → Lexical ”n” is deleted on the surface since the previous morpheme ends with a consonant
y:0 → Lexical ”y” is deleted on the surface since the previous morpheme ends with a consonant
H:0 → Lexical ”H” is deleted on the surface since the previous morpheme ends with a vowel
+:0 → Morpheme boundary
D:t → Lexical ”D” is realized as a surface ”t” since the previous morpheme ends with with a fricative consonant – one of ”ç,p,t,k,ş”, or ”D” is word final
D:d → Lexical ”D” is realized as a surface ”d” since the previous morpheme does

not ends with a fricative consonant – one of ”ç,p,t,k,ş”

Chapter 10
APPENDIX D - LEXICAL MORPHEMES

+lHk → FitFor/Ness

+lH → With

+ZHz → Without

+lAn → Acquire

+HmsH → JustLike

+ZAl → Related

+cH → Agent

+cHm → Short form of Diminutive for 1 person possessive

+cHk → Diminutive

+lAS → Become

+lAr → Plural/3rd person plural agreement

+lArarasI → InBetween

+sH → 3rd person singular possessive

+lArH → 3rd person plural possessive

+Hm → 1st person singular possessive

+Hn → 2nd person singular possessive

+yH → 3rd person singular possessive/Accusative case

+HmHz → 1st person plural possessive

+HnHz → 2nd person plural possessive

+nH → Accusative case

+yA → Dative case/Optative Mood

+nA → Dative case

+DAn → Ablative case

+ndAn → Ablative case

+DA → Locative case

+ndA → Locative case

+nHn → Genitive case

+nHm → Genitive case

+ylA → Instrumental case

+cA → Equative ly as in slow-ly

+ncA → Equative case

+ki → Relativized Pronoun/Modifier

+yDH → Past Tense

+ysA → Conditional Mood

+ymHS → Narrative Past Tense

+yken → While

+cAsHnA → AsIf

+ykene → While

+yHm → 1st person singular agreement

+ZHn → 2nd person singular agreement

+yHz → 1st person plural agreement/1st person plural agreement for negative aorist

+ZHnHz → 2nd person plural agreement

+DHr → Copula/Causative

+m → 1st person singular agreement

+n → 2nd person singular agreement

+k → 1st person plural agreement

+nHz → 2nd person plural agreement

+mHS → Narrative Past Tense

+DH → Past Tense

+sA → Conditional Mood

+HS → Reciprocal/Collective

+Hn → Reflexive/Passive

+Ar → Causative/Aorist

+Hr → Causative/Aorist

+Ht → Causative

+t → Causative

+Hl → Passive

+HnHl → Passive

+mA → Verbal Negation/Infinitive Marker

+yAmA → Verbal Negative Possibility Marker

+zsHn → 2nd person singular agreement for negative aorist

+z → 3rd person singular agreement for negative aorist

+zsHnHz → 2nd person plural agreement for negative aorist

+zlAr → 3rd person plural agreement for negative aorist

+yAbil → Able

+yAdur → Repeat

+yHver → Hastily

+yAgel → EverSince

+yAgOr → Repeat

+yAyaz → Almost

+yAkal → Stay

+yAkoy → Start

+yAgid → Continue

+mAksHzHn → WithoutHavingDoneSo

+mAdAn → WithoutHavingDoneSo

+yAmAdAn → WithoutBeingAbleToHaveDoneSo

+yHcH → Agent

+mAzlHk → NotState

+yAmAzlHk → NotAbleState

+mAK → Infinitive

+yHS → Infinitive

+mAcA → ActOf

+yAn → PresentParticiple

+yAcAk → Future Tense/Future Participle

+yAsH → FeelLike

+yAsHyA → Adamantly

+DHk → Past Participle

+DHkCA → AsLongAs

+yHncA → When

+yArAk → ByDoingSo

+yHp → AfterDoingSo

+yAlH → SinceDoingSo

+r → Aorist Mood

+Hyor → Present Continouns

+mAktA → Present Continouns State

+mAlH → Necessitative Mood

+ZA → Imperative Mood/Desiderative Mood

Bibliography

[1] *Oflazer K.:Two level description of Turkish morphology. Literary and Linguistic Computing 9 (1994) 137-148*

[2] *Fellbaum, C., ed.: WordNet, An Electronic Lexical Database. MIT Press (1998)*

[3] *Bilgin,O.,Çetinoglu, O., Oflazer, K.:Building a WordNet for Turkish. Romainian Journal of Information Science and Technology 7 (2004) 163-172*

[4] *Oflazer,K.,Inkelas, S.:The architecture and the implementation of a finite state pronunciation lexicon for Turkish. Computer Speech and Language (2006) Volume:20 No:1*

[5] *Koskenniemi, K., Two-level Morphology: A general computational model for word recognition and production. Publication No:11, Department of General Linguistics, University of Helsinki (1983)*

[6] *Nerbonne, J.: Computer assisted language learning and natural language processing. In Mitkov, R., ed.: Handbook of Computational Linguistics. Oxford University Press (2002)*

[7] *Nerbonne, J., Karttunen, L., Paskaleva, E., Proszeky, G.,Roosmaa, T.: Reading more into foreign languages. Proceedings of the Fifth Conference on Applied Natural Language Processing (1997)*

[8] *Borin, L. What have you done for me lately? The fickle alignment of NLP and CALL. In Proceedings of EuroCALL 2002 pre-conference workshop NLP in CALL (2002)*

[9] *Güvenir, H.A.: Drill and Practice for Turkish Grammar. In Swartz, M.L., Yazdani, M., eds: Intelligent Tutoring Systems for Foreign Language Learning. Volume F80 of NATO ASI Series. Springer Verlag (1992)*

[10] *Güvenir, H.A., Oflazer, K.: Using a corpus to teach Turkish Morphology. In proceedings of the Seventh Twente Workshop on Language Technology, Enschede, Netherlands (1994)*

[11] *Eclipse - an open development platform, http://www.eclipse.org*

[12] *Mozilla Firefox, http://www.mozilla.org*

[13] *Mysql - an open source database engine, http://www.mysql.org*

[14] *Java Technology, http://java.sun.com*

[15] *Apache Tomcat Application Server, http://tomcat.apache.org*

[16] *XMLHttpRequest object, base of the Ajax, specification by W3C. http://www.w3.org/TR/XMLHttpRequest/*

[17] *XUL-XML User Interface Language, http://www.mozilla.org/projects/xul*