MACHINE TRANSLATION FROM TURKISH TO OTHER TURKIC LANGUAGES

AND

AN IMPLEMENTATION FOR THE AZERI LANGUAGE

by

İlker Hamzaoğlu

B.S. in Computer Engineering, Boğaziçi University, 1991

Submitted to the Institute for Graduate Studies in

Science and Engineering in partial fulfillment of

the requirements for the degree of

Master of Science

in

Computer Engineering

Boğaziçi University

1993

MACHINE TRANSLATION FROM TURKISH TO OTHER TURKIC LANGUAGES

AND

AN IMPLEMENTATION FOR THE AZERI LANGUAGE

APPROVED BY:

Prof. Dr. Selahattin Kuru ....

(Thesis Supervisor)

Doç. Dr. Sumru Özsoy .....

Yrd. Doç. Dr. A. C. Cem Say ....

DATE OF APPROVAL 29.6.1993

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Prof. Dr. Selahattin Kuru for giving me the initial idea and for his encouragement during all the steps involved in the preparation of this thesis; Dr. Sumru Özsoy and Dr. Cem Say for taking time out of their extremely busy schedules to participate on thesis jury. I would also like to thank the following individuals. Tunga Güngör for his valuable comments about Turkish morphology; and the members of the Spelling Checker and Corrector project team Dr. Levent Akın, Tunga Güngör and Duygu Arbatlı for participating in the development of the Turkish morphological parser used in this thesis.

ABSTRACT

Machine translation is the application of computers to the translation of texts from one natural language into another. There are three different approaches proposed for machine translation; direct translation, transfer-based translation and interlingua-based translation. Since none of these approaches are suitable for the problem of machine translation from Turkish to other Turkic languages, we propose a lexicon-based approach. As the sentence syntax is similar for Turkish and the Azeri language, chosen as a representative of Turkic languages, we do not employ any syntactic analysis. Morphological and semantic analyses are carried out for the translation. Translation can not be viewed as a word for word translation even though the source and the target languages have a similar syntactic structure. This is due to the existence of ambiguous words with multiple meanings. The subject of the ambiguity in translation from Turkish to Azeri is explained and possible ways to resolve the ambiguities are put forward. A translator from Turkish to Azeri is implemented using the proposed approach. The results obtained from the translator show that the proposed approach is feasible for machine translation from Turkish to the Azeri language.

ÖZET

Makine çevirisi bilgisayarların doğal bir dilden bir diğerine metin çevirimine uygulanmasıdır. Makine çevirisi için önerilmiş üç değişik yaklaşım vardır; dolaysız çevirim, aktarma temelli çevirim, geçiş dili temelli çevirim. Bu yaklaşımların hiç biri Türkçeden diğer Türk dillerine makine çevirisi için uygun olmadığından biz sözlük temelli bir metot önerdik. Türkçe ve diğer Türk dillerinin temsilcisi olan Azeri dilinin sözdizimi yapıları benzer olduğundan sözdizimi analizi yapmadık. Çeviri için biçimbilimsel ve anlamsal analizler gerçekleştirildi. Kaynak ve hedef dillerin sözdizimi yapıları benzer olsa bile makine çevirisi kelimeden kelimeye olarak düşünülemez. Bunun sebebi birden çok anlamı olan belirsiz kelimelerdir. Türkçeden Azericeye çeviride belirsizlik konusu açıklandı ve bu belirsizlikleri çözmek için muhtemel yollar ortaya kondu. Önerilen yaklaşım tarzı kullanılarak Türkçeden Azericeye bir çeviri programı gerçekleştirildi. Çeviri programından alınan sonuçlar önerilen yaklaşımın Türkçeden Azericeye çeviri için uygulanabilir olduğunu gösterdi.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The motivation for *Machine Translation(MT)* is very clear. Translation among languages is economically and politically vital in the modern world. Translation among European languages, for instance, is a requirement for the European Community (EC). The EUROTRA project [1] of the Commission of European Countries for machine translation between European languages proves this argument with its grant of 16 million ECU (about 12 million dollars) by the Council of Ministers for a five and a half year program of research and development. The U.S. and Japanese economies rely on export markets in a large number of languages, where English alone does not suffice. The emergence of new Turkish Republics in Central Asia following the collapse of the former Soviet Union created a potential market for machine translation between Turkish and the other Turkic languages.

Scientists, technologists, engineers, economists, agriculturalists, administrators, industrialists, businessmen and many others have to read documents and have to communicate in languages that they do not know. In addition to the high cost and unavoidable delays of human translation, human translators are also unable to cope with the ever increasing volume of material which has to be translated. With this indisputable need for massive, timely and inexpensive translation, the dream of many computational linguists and computer scientists has been to achieve fully automated machine translation.

Translation between Turkic languages is a new area possibly with interesting problems, e.g. seeing if conventional approaches will work for the problem, investigating different sorts of ambiguities in Turkish, investigating appropriate lexicon structures, and seeing how the close relationships among the Turkic languages can contribute to reduce the complexity of translation process. Moreover, *Natural Language Processing(NLP)* of Turkish is an emerging area in Turkish NLP research communities, and translation system is a good testbed for NLP applications.

In this respect we studied the problem of machine translation from Turkish to other Turkic languages. Since Turkic languages are agglutinative languages, we have actually worked on the problem of translation from an agglutinative language to another one. Although there is a considerable amount of literature on the problem of machine translation, almost none is related with this problem. To the best of our knowledge this work has been the first attempt in this field.

This thesis discusses the problem of machine translation from Turkish to other Turkic languages. Up to now, three major approaches were proposed in the literature for machine translation, direct translation, transfer-based translation and interlingua-based translation. These approaches differ in the scope of translation they cover. Since none of these traditional approaches is suitable for our problem, we propose a lexicon-based approach to achieve the translation. We have compiled those techniques of the conventional approaches that can be utilized in this approach and incorporated them into the approach.

As the sentence syntax is similar for Turkish and the Azeri language, representative of other Turkic languages, we do not employ any syntactic analysis. Even though the source and the target languages have similar syntactic structure, translation can not be viewed as a word for word translation. This is due to the existence of ambiguous words conveying multiple meanings. The subject of ambiguity in translation from Turkish to Azeri is investigated, different sorts of ambiguities and lexical data that cause those ambiguities are identified, and possible ways to resolve them are put forward. A practical machine translation system for translation from Turkish to Azeri is developed using the proposed approach.

The second chapter includes the discussion of machine translation, and practical machine translation systems developed for Turkish. The third chapter discusses the problem of morphological parsing for agglutinative languages which is especially critical for this thesis work. Chapter 4 presents the characteristics of Turkish and the Azeri language from

translation perspective and discusses the approach proposed for solving the problem of machine translation from Turkish to Azeri. Chapter 5 describes the implementation of the translator in terms of the contents and structures of its lexicons, and the algorithms. Chapter 6 puts forward the shortcomings of the translator, its performance evaluation and further improvements. The last chapter gives an idea of further developments and summarizes the work accomplished in a final conclusion section.

Throughout the thesis, all the Turkish and Azeri expressions are given in italics, and the meaning in English immediately follows in brackets, e.g. *ayak (foot)*. The suffixes are given together with the symbol -, and capital letters and the symbol () are utilized to represent the allomorphs of the suffixes as explained in Section 4.1.2, e.g. -(y)A (dative). The symbol - coming just after a Turkish word represents a verb, e.g. *gel-(to come) and koş-(to run)*. Since, in this thesis we deal with machine translation from Turkish to Azeri, the terms "translation" and "machine translation from Turkish to Azeri" are going to be used interchangeably.

2. MACHINE TRANSLATION

Machine translation is the application of computers to the translation of texts from a natural language into another. There are three different approaches proposed for MT. Machine translation and the proposed approaches are discussed in this chapter.

2.1. Machine Translation within the Context of Natural Language Processing

Natural language processing is an area of artificial intelligence that deals with automating the various language-oriented tasks that are currently performed predominantly by humans. The domain of NLP includes applications such as *machine translation, language understanding, language generation, natural language interface to databases and expert systems, morphological and syntactic parsing, spelling checking, grammar checking and thesaurus.*

One of the first linguistic applications of computers to be envisaged and funded was machine translation. The general task of machine translation can be described very simply as follows [2]:

> "Feed a text in one language (SL, for source language) into a computer and, using a computer program, produce a text in another language (TL, for target language) such that the meaning of the TL text is the same as the meaning of the SL text."

The history of machine translation goes back to late 1940s. Starting with the so-called Weaver Memorandum [2] which was claiming the feasibility of MT, different scientific groups in Europe and USA worked on the field. Through the 1950s and into the following decade, research in MT continued and grew.

The first generation machine translation programs were constructed on rudimentary linguistic theories. They ignored the fact that meaning was essentially involved in the translation process, and employed word-by-word substitution to achieve the translation. Because of these reasons, their results were disappointing. This caused the release of famous ALPAC (Automatic Language Processing Advisory Committee) report [2] in USA in 1966, evaluating the practicality of contemporary MT research. The negative comments of this report drastically reduced the level of support for MT research.

Together with the failure of first generation machine translation programs, it was realized that machine translation is not a trivial process. It needs well-formed linguistic theories and supporting products provided by other natural language processing applications such as morphological analysis, syntactic analysis, semantic analysis, language understanding and language generation.

The most obvious deficiency of any word for word translation is that the order of words in the resulting target language text is more often wrong than correct. The solution of this problem demands some kind of syntactic analysis of the input texts.

The second problem is that there are rarely one to one correspondences in the vocabularies of natural languages. In most cases, a particular SL word may correspond to a number of different TL words. MT systems either print out all the possibilities or attempt to select the one which is most appropriate for the specific text under translation. Some kind of morphological, syntactic or semantic analysis or a combination of these should be taken into account to resolve this kind of ambiguities.

The number of ambiguous words in the source language text can be easily restricted by choosing a sufficiently narrow subject domain. This attitude in MT is known as the *sublanguage approach*. In this way, instead of the general language, MT translation systems are designed for a specified subfield of the language.

Despite the ALPAC report, considering these facts MT research and development continued in Europe and later in Japan. Notable MT achievements, with the SYSTRAN and EUROTRA projects in Europe and the Mu project at Kyoto University in Japan, gradually led to a revival of MT research in USA.

Until very recently, most MT systems necessitate postediting, improvement of the results of machine translation by a human. To decrease the postediting requirement, some MT systems allow human interaction during translation process, which is known as *human assisted machine translation*. One of the main research and development objectives in this field is to enhance the power of MT systems so that the extent of postediting is reduced and, eventually the need for it is eliminated.

Together with the spectacular advances in computer technology and computational linguistics, MT researchers and supporters are now optimistic about reaching the ultimate goal of machine translation, constructing a fully automated machine translation system.

Although the major objective of MT researchers is to achieve fully automated machine translation, many of the methods, techniques, and tools of machine translation can be used to build sophisticated tools for human translators, e.g. terminology banks, spelling and grammar checkers, text production facilities. Human translation with the use of these sophisticated tools is known as *machine aided translation.*

In machine translation there have been several efforts to develop procedures for evaluating the quality of the results produced by machine translation systems. Some quality and performance measures were proposed for this purpose. The following is a set of these performance metrics [2] :

1. *Linguistic generality*: the number of source and target languages covered by the system and the extent of coverage in the grammar and the general vocabulary.

2. *Application-domain generality*: the number of subject domains covered by the system and the extent of coverage of each domain.

3. *Degree of automation*: the extent to which human must intervene in the translation.

4. *Semantic accuracy*: the degree to which the translated text expresses the same meaning as the source text.

5. *Intelligibility*: the degree to which the translated text is easily understandable by readers of the target language without access to the source text.

6. *Appropriateness*: the degree to which the target text is stylistically appropriate for its intended audience.

7. *Domain and language portability*: the ease with which additional subject domains and languages can be added.

8. *Extensibility*: the degree to which an MT system provides for seamless and incremental extensions to the grammatical and lexical coverage of the languages and subject domains already in the system.

9. *Improvability*: the degree to which a system permits changes and enhancements to the level of automation as domain or lexical knowledge improves, without significantly compromising the quality of the translation.

10. *Ergonomics*: the extent to which the user interface provides maximum communication bandwidth, maximal clarity and minimal opportunity for errors.

11. *Integrability*: the extent to which an MT system can be an integral component of a complete authoring and document-production facility.

12. *Software portability*: the ease with which the MT software can be ported to other hardware platforms, other operating systems and so on.

2.2. Methods of Machine Translation

Machine Translation methods fall into three major categories [1,2,3] :

1. direct translation
2. transfer-based translation
3. interlingua-based translation

The central point of difference among them is on the depth of source and target language analysis. These methods are discussed below.

2.2.1 Direct Translation

Direct MT systems are the first generation machine translation systems [1,3]. They translate from a specific source language to a specific target language without any intermediate steps as shown in Figure 2.1.

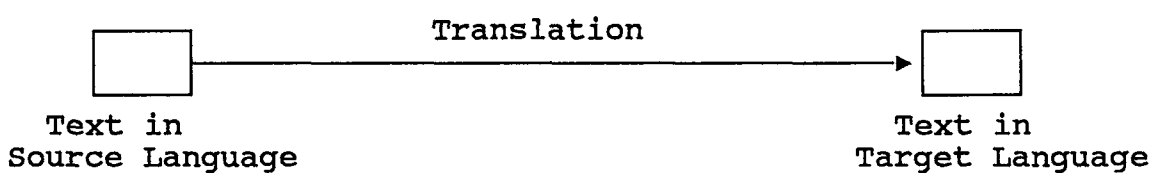

Figure 2.1. Direct Translation

They realize word for word translation relying on the assumption of finding direct correspondences between all the lexical items of the source and the target language. They assume that the SL texts need not be analyzed any more than strictly necessary for resolution of ambiguities. They have been highly criticized for their ad hoc techniques to resolve the ambiguities and they lost their scientific standing.

Although some of the current commercial MT systems, notably SYSTRAN, were built using the direct MT technology, they are currently being converted to employ the other approaches.

2.2.2 Transfer-based Translation

Transfer-based MT systems also translate from a specific language to another specific language [1,2,3]. However, they accomplish the translation process in three steps as illustrated in Figure 2.2.

1. source language analysis
2. source-to-target language transfer
3. target language generation

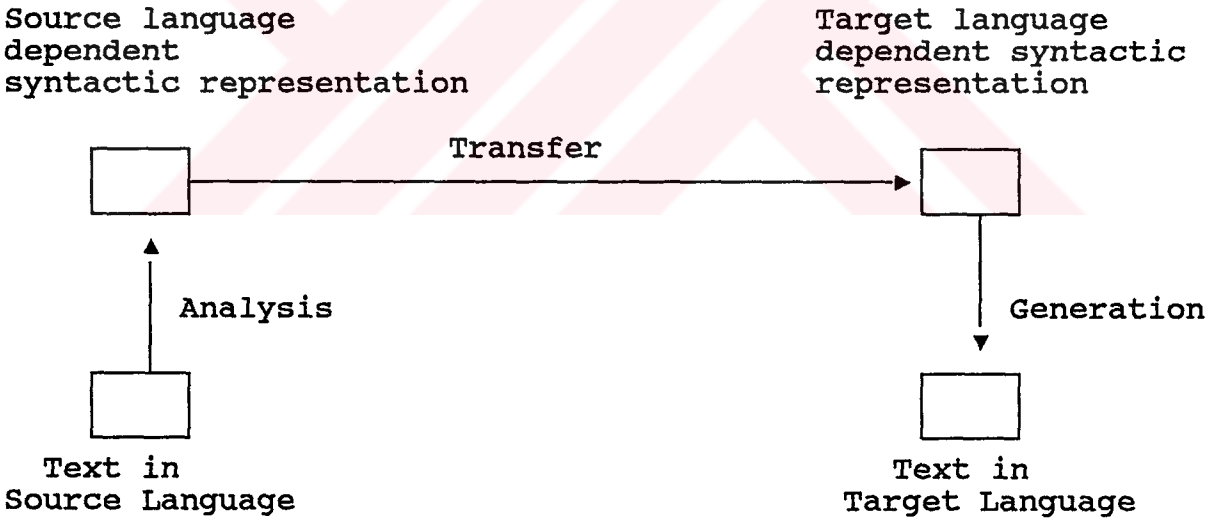

Figure 2.2. Transfer-based Translation

In the first step, syntactic analysis is performed on the source language text to transform it into a syntactic representation using the source language lexicon and the grammar. Then, source language dependent syntactic representation is transferred into a corresponding syntactic representation specific to the target language. This stage of

transforming standard syntactic structures is implemented through a bilingual lexicon and the grammars of both languages. In the last step, target language text is produced from the syntactic representation of the target language by utilizing the target language dictionary and the grammar.

In the transfer approach only those ambiguities inherent in the source language are tackled in source language analysis. Differences between languages are handled during transfer.

2.2.3 Interlingua-based Translation

Transfer systems are not suitable for multilingual translation, since a different transfer unit is necessary for each pair of languages. If there are n languages involved and translation is to be from and into each of them, then a system would need n(n-1) binary transfer units. The solution of this problem is to separate the source and the target languages in the translation process. This idea is the basis of *interlingua approach* [1,2,3]. In the interlingua approach, the translation system for n languages would need just 2n transfer units, since source and target languages are never in contact. Interlingua systems basically involve two stages for translation as shown in Figure 2.3.

1. source language analysis
2. target language generation

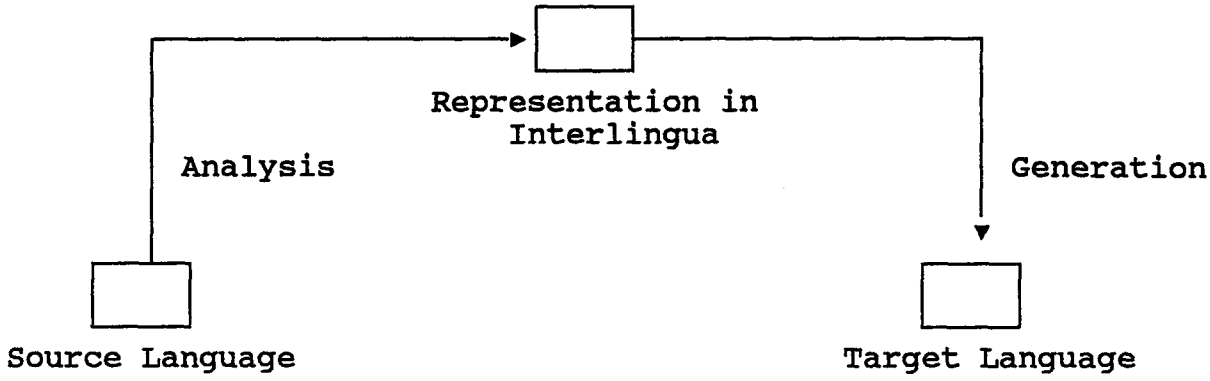

Figure 2.3. Interlingua-based Translation

In the first stage the meaning of the source language text is represented in an unambiguous formal artificial language, interlingua. Then in the second stage this meaning is expressed in target language using the lexical units and syntactic constructions of the target language. Procedures for source language analysis are intended to be SL specific and not devised for any particular TL in the system. Similarly, target language generation is intended to be target language specific.

The major distinction between the interlingua-based and the transfer-based systems is the attitude toward comprehensive analysis of meaning. Transfer-based systems achieve translation without deep understanding of the source language text. However, interlingua approach necessarily requires complete resolution of all ambiguities and anomalies of SL texts so that translation should be possible into other languages.

The most famous example of interlingua-based machine translation systems is the Eurotra project [1]. Eurotra, started in 1978, is an ambitious, well-supported project aiming to provide MT among all official European Community languages (Danish, Dutch, English, French, German, Greek, Italian, Portuguese and Spanish) . The project is still under development, and it represents a mile stone in MT research.

2.2.3.1 Knowledge-based Translation

In the last few years, *knowledge-based approach* is proposed as a descendant of interlingua oriented machine translation [2]. This new approach is developed in the Center for Machine Translation at Carnegie Mellon University. The central principle underlying this approach is the stress on functionally complete understanding of the meaning of the source text as a prerequisite to successful translation. Thus, it requires a much deeper level of source language analysis in order to be able to translate between a number of languages.

Since it belongs to the class of interlingua-based systems, translation within knowledge-based systems is also a two step process, analysis and generation.

The main difference between the interlingua-based approach and the knowledge-based approach is on the depth of source language analysis and the reliance of knowledge-based systems on explicit representation of world knowledge. Knowledge-based machine translation systems must be supported by world knowledge. They utilize a model of the world, *an ontology*, including knowledge about basic types of objects and events in the physical world, e.g. *a car is a kind of vehicle*, relationships among them, e.g. "is-a" , "part-of", and particular instances of object types, e.g. IBM is an instance of the object type "corporation", and election of Bill Clinton as president of the United States is an instance of the complex action "to_elect".

With special attention paid to acquisition of large knowledge bases and with the advent of new tools the practicality of the knowledge-based approach is growing steadily. Since a totally comprehensive analysis of meaning is not yet feasible, and the attainment of this goal will remain an objective of computational linguistics for years to come, a practical knowledge-based system will attain a lesser depth of understanding.

2.3. Machine Translation Research on Turkish

Up to now, there have been several works on the machine translation problems involving Turkish either as a source language or as a target language. We examined the following two works of this kind.

The work of Özgüven and Tsujii [3] is about MT from English to Turkish restricted in the domain of the news reports sublanguage. They employ a kind of transfer-based method for translation. The translation is carried out by translating phrases as a whole, and then assembling them. The translator first locates the phrases in the input text with Phrase Analysis module, and then translates each phrase with the Phrase Translation module. Finally, they form the Turkish sentences by assembling the translations of phrases in the correct syntactic structure with the Structural Translation module. The major shortcoming of the translator is the fact that they don't attempt to resolve the lexical and structural ambiguities. Although they state that the results obtained from the translator were good, they also remind that these results were produced by only a small model of the actual design.

Another work on the problem of machine translation involving Turkish is the Transit (Translation System Into Turkish) project started in September 1985 in the Department of Computational Linguistics of the University of Nijmegen [4,5]. It is a large scale project aiming to achieve machine translation from Dutch to Turkish. They utilize transfer-based method. In this respect, they develop several programs to realize each phase of the translation. The programs called AMAZON and CASES are responsible for the source language analysis. They syntactically and semantically analyze the Dutch sentences and provide the semantic representations for them. On the other hand, programs called AMATUMOR (Automatic Turkish Morphological Analyzer), AMATURKA (Automatic Turkish Syntactic Analyzer) and ATMACA (Semantic Analyzer for Turkish) are responsible for the same processes for Turkish. They are currently working on the integration of these two groups of by-products to achieve the

translation.

As far as we know no results are published on their translator yet. However, the results obtained from ATMACA are available. Their Turkish morphological analyzer AMATUMOR determines the morphological ambiguities and ATMACA tries to resolve them by utilizing semantic features.

Since Turkish and its dialects are agglutinative languages, we have actually worked on the problem of translation from an agglutinative language to another one. The translation systems explained above do not involve agglutinative languages (see Chapter 3) as both the source and the target language. In both systems only the target language, Turkish, is agglutinative. To the best of our knowledge, there is no work done on machine translation research where both the source and the target languages are agglutinative, and on the problem of machine translation from Turkish to other Turkic languages. Therefore this work is the first attempt on machine translation from Turkish to other Turkic languages as well as from an agglutinative language to another one.

3. MORPHOLOGICAL PARSING

Morphological parsing is an area of natural language processing as stated in Section 2.1. It is utilized to determine the constituents of a word, its root and affixes.

3.1. Morphological Parsing in Agglutinative Languages

Morphological parsing is especially critical for *agglutinative languages* like Turkish, Finnish, Hungarian, Quechua and Swahili.

In agglutinative languages words are combinations of several morphemes. There is a root and several suffixes are affixed to this root to form new words, either modifying or extending its meaning. In agglutinative languages stem formation by affixation to previously derived stems is extremely productive. A given stem, even though itself quite complex, can serve as a basis for even more complex words. Consequently, agglutinative languages contain words of considerable complexity, and parsing such languages requires a detailed morphological analysis.

The phonemes and morphemes are two basic constituents of the morphology in agglutinative languages. Their definitions are as follows. A *phoneme* is the unit of sound, and *allophones* of a phoneme are its variant forms as conditioned by position or adjoining sounds, e.g. the allophones of the Turkish phoneme **ı** are **ı, i, u, ü.** A *morpheme* is the smallest unit of speech bearing a meaning, and the *allomorphs* of a morpheme are its different forms it might take according to the context it appears, e.g. the allomorphs of the Turkish morpheme -lAr are -lar and -ler.

In agglutinative languages, morphotactic rules determine the way morphemes are ordered to form a word. A given morpheme may take a shape, one of its allomorphs, dependent on its morphological and phonological environment. The morphophonemic alternation rules determine the surface form of the morpheme, namely its allomorph.

Morphological analysis methods fall into two major categories:

1. Listing methods
2. Computational methods

Listing methods are not suitable for agglutinative languages as discussed by Hankamer[6] and Kibaroğlu[7]. Computational methods are divided into two categories:

1. Left-to-right parsing (Root matching method)
2. Right-to-left parsing (Suffix stripping method)

Left-to-right parsing is more suitable for agglutinative languages than right-to-left parsing, as stated by Hankamer[6] and Kibaroğlu[7].

In the computational methods, there are two representations of the language, lexical and morphological. While lexical representation includes the vocabulary of the language, morphological representation contains the rules of the language which determine the way morphological compounds come together.

In left-to-right parsing, the parser employs a finite state transition network representation of morphotactics and a treatment of morphophonemic alternations. Parsing proceeds as follows. Roots are sought in the lexicon that match initial substrings of the word, and the grammatical category of the root determines what class of suffixes may follow. When a suffix in the permitted class is found to match a further substring of the word, grammatical information in the lexical entry for that suffix determines once again what

class of suffixes may follow. If the end of the word can be reached by iteration of this process, and if the last suffix analyzed is one which may end a word, the parse is successful.

3.2. Research on Morphological Parsing of Turkish

Morphological parsing has attracted relatively little attention in computational linguistics until recently. The reason seems to be the fact that virtually all parsing research has been concerned with English, or with other languages morphologically similar to English. Since in these languages words contain only a small number of affixes or none at all, almost all of the parsing models for them assume either that there is no real need for morphological parsing or that any such morphological parsing, recognizing this small number of affixes, will be trivial.

However, as explained in above section, this is not the case with agglutinative languages, hence for Turkish. Agglutinative structure of Turkish requires an extensive analysis of its morphology to build a morphological parser for parsing Turkish words. So far, several researchers worked on the problem of morphological parsing of Turkish and produced morphological parsers for parsing Turkish words. Kibaroğlu[7], Akın et. al.[8], Hankamer[9] and Solak and Oflazer[10] are among those researchers. In all of these works, the parsers employ left-to-right parsing method. The morphological parser produced in the project "A Spelling Checker and Corrector for Turkish" [8] is adapted and used in the translator developed in this work.

4. MACHINE TRANSLATION FROM TURKISH TO OTHER TURKIC LANGUAGES

This chapter discusses machine translation from Turkish to other Turkic languages. Since Turkic languages are agglutinative languages, this discussion is actually about the problem of machine translation from an agglutinative language to another one.

Azeri Language is chosen as a representative of the other Turkic languages, since it is closer to Turkish more than all the others. The example in Table 4.1 illustrates this similarity [11]. Note that capital letters represent characters not available in the Latin alphabet.

Table 4.1. The appearance of a sentence in Turkic languages

Sentence	Language
abla-sı Yıldız-la telefon-da konuş-uyor.	(Turkish)
böyük bacı-sı Yıldız-la telefon-da danış-ır.	(Azeri Language)
eceke-si Yıldız bilen telefon-da gürleş-Yar.	(Turkmen Language)
apa-si Yıldız bilAn telefon-Da sözlAş-AyApti.	(Uzbek Language)
apa-sı Yıldız-ben telefon-da söyleş-ip tur.	(Kazakh Language)
ece-si Yıldız menen telefon-da süylöş-up cattı.	(Kırghız Language)
apa-sı Yıldız bilAn telefon-da söylAş-A.	(Tatar Language)

Of the three major approaches to machine translation, namely direct translation, transfer-based translation and interlingua-based translation, none fits this problem alone. So we propose a lexicon-based approach to achieve the translation.

4.1. Turkish and The Azeri Language

In this section we discuss the key features of Turkish and the Azeri Language. Turkish is an agglutinative language in which syntactic relations between words are expressed through discrete suffixes. Turkish is a subject-object-verb language, however the order of phrases may be changed to emphasize certain constituents of the sentence. The main constituent of a Turkish sentence is verb. The usage of other constituents is optional and dependent on the properties of verb [12]. Azeri is also an agglutinative language with a similar syntactic structure. Thus the discussion is given in terms of the alphabets, the lexicons, the phonology and the morphology.

4.1.1. Alphabet, lexicon and phonology

Turkish uses an alphabet of 29 letters in its current orthography using Latin characters. There are 8 vowels: **a, e, ı, i, u, ü, o, ö,** and 21 consonants: **b, c, ç, d, e, f, g, ğ, h, j, k, l, m, n, p, r, s, ş, t, v, y, z.** On the other hand, Azeri has 9 vowels and 23 consonants. It has the vowel **ä**, and the consonants **x** and **q**, in addition to letters of the Turkish alphabet [13,14]. Azeri has many words common with Turkish. About 1100 of the 6900 Azeri words, approximately 1/6, in the "Comparative Dictionary of Turkish Dialects" [13] are the same with their Turkish equivalents. Azeri also has many words borrowed from various other languages such as Greek, English, and Russian. Velosiped(bicycle), lampoçka(lamp), prodyüser(producer), prospekt(street), and respublika(republic) are examples of those words.

Some Azeri words are phonological variations of Turkish words. The domain of some phonological variation rules covers all the Azeri words, whereas some of them can influence only a group of Azeri words. The phonological variation rules valid for all the Azeri words are the following:

1.the consonant **k** at the end of the polysyllabic words corresponds to **g** if the last vowel of the word is in the set {a,ı,o,u}.

2.the consonant **k** at the beginning of the words corresponds to **g** if the first vowel of the word is in the set {a,ı,o,u}.

About 190 of the first 1100 Azeri words, approximately 1/5, in the "Comparative Dictionary of Turkish Dialects" [13] obey the second type of rules. The phonological variation rules valid only for some Azeri words are the following:

1. the vowels **a, e, i** correspond to **ä.**

2. the vowels **u, ü** correspond to **o, ö,** respectively.

3. the consonant **y** at the beginning of the word is deleted and the following consonant **ı** corresponds to **i** or **u.**

4. the vowel **ı** at the beginning of the word corresponds to **i.**

5. the consonant **h** is attached to the words beginning with **ü** or **ö.**

6. the vowels **a** and **e** correspond to **o** and **ö,** respectively, if they come just before the consonant **v** in the word.

7. the consonant **k** corresponds to **x.**

8. the consonants **ş, d, g, k** and **y** correspond to **şş, dd, gg, kk** and **yy,** respectively.

9. the consonant **k** at the end of the word corresponds to **y.**

10. the consonants **ğ, v** between two vowels corresponds to **y.**

11. the consonants **pr** and **vr** correspond to **rp** and **rv,** respectively.

12. the consonants **p, ç, t** and **k** at the end of the word correspond to **b, c, d** and **g,** respectively.

4.1.2. Morphology

As agglutinative languages Turkish and Azeri use derivational and inflectional suffixes to form new words. Derivational suffixes alter the meaning of the word, whereas inflectional suffixes do not.

There is a one-to-one correspondence between the inflectional suffixes of Turkish and Azeri, either in the same form or in a different form, except the following two:

1. The Turkish suffix -mIş forming past indefinite tense has two equivalents in Azeri, -mIş and -Ib, e.g. the meaning of the Turkish word *gelmiş* can be expressed with either *gälmiş* or *gälib* in Azeri.

2. The Turkish suffix -mAlI forming the necessitative modal has two equivalents in Azeri, -mAlI and -AsI. The word *gärək* together with the suffix -(y)A is also used for this modal, e.g. the meaning of the Turkish word *gelmeli* can be expressed in three ways in Azeri: *gälmäli, gäläsi*, and *gärək gälə*.

Since, for both tenses, all the equivalent Azeri suffixes give the same meaning, only one of them is considered in the translation process.

However, the same is not true for derivational suffixes. Some of the derivational suffixes of Turkish do not exist in Azeri. The suffixes -sAl, -giller, -ölçer and -(i)k are the examples of this kind of suffixes. Moreover, some of them which have equivalents in Azeri, may not be applicable to all Azeri words. The suffixes -lA, -kâr and -hane exemplify this kind of suffixes.

This situation for the suffix -lA (derivational suffix from noun to verb) is illustrated in Table 4.2. For instance, the Turkish word *bağ (string)* can be used as a stem in the derivation of the word *bağla- (to tie)* with the suffix -lA. Since the Azeri equivalent of *bağla-* can be derived from the Azeri equivalent of *bağ* with the suffix -lA, the suffix is

applicable for this word. However, the same is not valid for the Turkish words *sergi (exhibition)* and *sergile- (to exhibit)*. Although *sergile-* can be derived from *sergi* with the suffix -lA in Turkish, its Azeri equivalent *särgiyä goy-* can not be derived from the Azeri equivalent of *sergi* with the suffix -lA. This implies that the suffix -lA is not applicable for this word.

Table 4.2. The usage of the suffix -lA in the Azeri Language

status	Turkish word	Azeri equivalent
applicable	bağ(string) bağla-(to tie)	bağ bağla-
applicable	dilim(slice) dilimle-(to slice)	dilim dilimlä-
applicable	hesap(calculation) hesapla-(to calculate)	hesab hesabla-
not applicable	harman(blend) harmanla-(to blend)	xırman garışdır-
not applicable	sergi(exhibition) sergile-(to exhibit)	särgi särgiyä goy-

Turkish has the vowel harmony rule and the consonant harmony rules defining the morphophonemics of the language and some morphotactic rules determining the orders of morphemes [7,8,10].

Vowel harmony is a process by which the vowels in all syllables of a word except the first assimilate to the preceding vowels with respect to certain phonetic features. The Turkish vowels **a, e, o, ö** are classified as low vowels, and the ones **ı, i, u, ü** as high vowels. Except the present tense suffix -Iyor, there are no suffixes in which the low vowels **o** and **ö** appear. Therefore in citing suffixes, the letter A is used for low vowels and I for high vowels. When a suffix is affixed to a stem, the first vowel in the suffix changes according to the last vowel of the stem. Succeeding vowels in the suffix change according to the vowel preceding them. So the two classes of vowels are resolved as follows:

A = a if the previous vowel is in the set {a, ı, u, o}
= e if the previous vowel is in the set {e, i, ü, ö}

I = ı if the previous vowel is in the set {a, ı}
= i if the previous vowel is in the set {e, i}
= u if the previous vowel is in the set {o, u}
= ü if the previous vowel is in the set {ö, ü}

The following examples illustrate the usage of vowel harmony rule. *gel (to come)* + -DI (past definite tense) = *geldi*, *koş (to run)* + -Iyor (present tense) = *koşuyor*, and *defter (notebook)* + -lAr (plural suffix) = *defterler*.

Because of their different phonetic structures, some words borrowed from other languages do not obey vowel harmony rule during agglutination, e.g. *saat (watch)* + -(y)A (dative) = *saate*.

Besides vowel harmony rule, there exist another morphophonemic rule involving vowels of the morphemes, which is known as *elision*. Morphemes beginning with a vowel are affixed to stems ending with a vowel with the deletion of their first vowel, e.g. *masa (table)* + -(I)m (1st singular possessive) = *masam*. The first vowels of these suffixes are shown with the symbol (), as in this example.

Turkish consonants are classified in two sets: the consonants **ç, f, t, h, s, ş, k, p** as voiceless consonants and **b, c, d, g, ğ, j, l, m, n, r, v, y, z** as voiced consonants. The consonant harmony rules listed below are based on this classification [12,15,16].

1. *consonant mutation rule*: In multi-syllabic words and in certain mono-syllabic roots, the final voiceless consonants **p, ç, t, k** are mostly changed to **b, c, d, g,** respectively, when a suffix beginning with a vowel is attached to the word, e.g. *kitap (book)* + -(y)I (accusative) = *kitabı*.

2. *consonant assimilation rule*: In some suffixes beginning with one of the consonants **c, d,** or **g,** this consonant might change to **ç, t, k,** respectively, if the last phoneme of the stem to which one of such suffixes is attached is a voiceless consonant, e.g. *gel (to come)* + -DI (past definite tense) = *geldi*, but *koş (to run)* + -DI (past definite tense) = *koştu*. These consonants are shown as C, D and G, as in the example.

3. *phoneme insertion rule*: Some morphemes beginning with a vowel are affixed to the stems ending with a vowel with the insertion of one of the consonants **n, s,** or **y,** e.g. *ev (house)* + -(s)I (3rd singular possessive) = *evi*, but *kapı (door)* + -(s)I (3rd singular possessive) = *kapısı*, and the consonants inserted are shown as (n), (s), and (y).

Azeri has similar morphotactic rules with Turkish, but not the same morphophonemic rules. Vowel harmony rule exists in Azeri including the vowel **ä**. The vowel **e** is not used in Azeri suffixes, and the vowel **ä** is treated the same as the Turkish vowel **e** in the application of the vowel harmony rule.

Consonant harmony rules exist in Azeri with some modifications. Since the voiceless consonants **p, ç, t, k** already appear as **b, c, d, g** as the last letters of Azeri root words, the first consonant harmony rule is not utilized in the Azeri. The second consonant harmony rule is not valid for the consonant **d,** which is always used as **d.** However it is valid for the consonants **c** and **g** with the following two exceptions. The derivational suffixes -CI, -CIk deriving nouns from noun root words always appear as -çI and -cIK, respectively. The phoneme insertion rule is the same with the Turkish equivalent with the exception that, instead of inserting letter **y** letter **n** is inserted as auxiliary letter with accusative suffix.

Azeri has an additional consonant harmony rule that does not exist in Turkish. The consonant **k,** which will be shown as K, in the suffixes changes according to its preceding vowel. It takes one of its allophones **k** or **g** according to the following rule:

K = g if the previous vowel is in the set {a, ı, u, o}

= k if the previous vowel is in the set {ä, e, i, ü, ö}

4.2. Translation From Turkish to The Azeri Language

None of the approaches discussed in Chapter 2 is suitable for machine translation from Turkish to the Azeri language. Since translation from Turkish to Azeri requires a detailed morphological analysis and detailed processing for ambiguous words, direct translation alone is not suitable for this purpose, as it is not for many other translation problems. Transfer based approach requires a detailed syntactic analysis. Since the syntactic structures of Turkish and Azeri are similar, there is no need for such a detailed syntactic analysis for this problem. The aim of the interlingua-based approach is to achieve multilingual translation among completely different languages and this is quite different from this problem. Therefore the construction of a formal interlingua language to express the meaning is redundant for our problem. Knowledge-based approach requires a much deeper analysis to get the complete meaning of the input text, and it is also redundant for our problem. Therefore we have proposed a lexicon-based approach and incorporated some of the techniques used in conventional approaches into it in order to solve the problem of translation from Turkish to other Turkic languages.

With this approach we claim that translation from Turkish to Azeri can be achieved without completely understanding the input text, and the translation process can be viewed as a word for word translation of sentences. Because the number of words that can be translated into Azeri with one-to-one correspondence of the words is much larger than the number of ambiguous words that needs additional processing to be translated correctly. We determined 750 of 6900 Turkish words in our root lexicon as ambiguous in the context of machine translation from Turkish to Azeri. Therefore representing the meaning of each Turkish word seems unnecessary for this problem. Only the ambiguous words that should be disambiguated in the translation process are taken into account.

The ambiguities that do not cause any problem are ignored in the translation. All the information for the ambiguity resolution process is stored in the lexicons, and the disambiguation is carried out in a lexicon-based manner.

Moreover, as the sentence syntax is similar for both languages, there is no need to employ syntactic analysis to find out the syntactic structures of the Turkish sentences. However, we note that syntactic analysis may help to resolve the ambiguities in the semantic level. Even though we did not employ syntactic analysis, we utilize some of the syntactic rules of Turkish in resolving ambiguities.

Thus, we get the idea of word for word translation from direct translation approach, the techniques for morphological and semantic analyses from the transfer-based and interlingua-based translation approaches. Translation is achieved in 4 steps :

1. morphological analysis
2. semantic analysis
3. resolving the ambiguities
4. word generation by replacing the Turkish root word and the suffixes with their Azeri equivalents

Translation starts with the morphological analysis phase. At the end of the morphological analysis, all possible parses for a single word are obtained. For each parse of the word the Azeri equivalent of the root word and the suffixes involved in the parse are also kept. Semantic analysis is performed concurrently with morphological analysis, and the necessary information to be used in a possible disambiguation process is obtained. The characteristics of this information is explained in Section 4.5. Then the lexical item is disambiguated using the semantic information if there exists any ambiguity. Finally, the word is translated into Azeri by replacing the Turkish root word and the suffixes with their Azeri equivalents.

During the translation if a word can not be morphologically parsed in any way, it means that the word does not exist in the translator's lexicons. In this case the Turkish word is given as the result of the translation together with the message "No Information". Proper nouns are translated into Azeri without any modification.

4.3. No Ambiguity Case

In the simplest case no ambiguity exists. The word can be parsed in just one way and the root word has just one equivalent in Azeri. In this case, the word is translated into Azeri directly by replacing the root word and the suffixes with their Azeri equivalents. For instance, the Turkish word *konuşuyorum (I am speaking)* has a single parse:

konuş(to speak) - uyor(present tense) - um(1st single person)

and it is translated into Azeri by a one-to-one substitution as *danış-ır-am*.

As another example consider the Turkish sentence *arabayı yıkadı (He/she washed the car)*. The result obtained from the morphological parsing of this sentence is:

araba(car) - yı(accusative) yıka(to wash) - dı(past definite tense)

Since no ambiguity exists, it is directly translated into Azeri as *araba-yı yu-du.*

4.4. Ambiguity in Translation

With the disappointing results of the first generation machine translation systems, it is literally accepted that machine translation can not be considered as word for word translation without any additional processing. As discussed in the previous sections,

machine translation in general involves three levels:

1. morphological analysis
2. syntactic analysis
3. semantic analysis

In each of these analyses, there exist cases in which the analysis yields multiple solutions. The major problem in machine translation is the selection of the correct solution in these ambiguous cases.

In the case of syntactic analysis, ambiguity exists when a sentence or phrase can be represented with more than one syntactic structure, which is called *parse tree*. The following example illustrates this fact [17]. Turkish noun phrase *sıcak et reyonu satıcısı (the hot/warm meat stand seller)* can be interpreted in three ways as shown in Figure 4.1, and additional analysis is needed to identify the correct one.

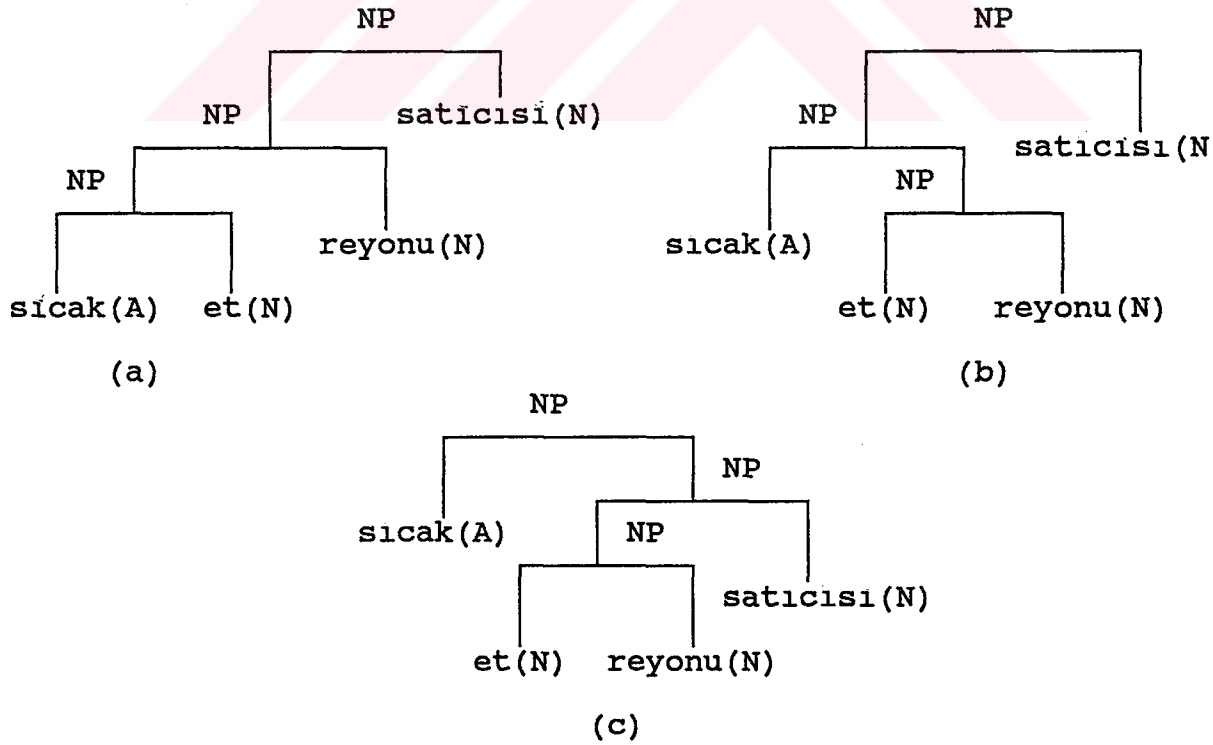

Figure 4.1. Three ambiguous interpretations of the noun phrase *sıcak et reyonu satıcısı*

In this case the problem is to identify the modifiers and the way they modify the other nouns in the noun phrase. However, this does not introduce a problem in translation from Turkish to Azeri, since the sentence syntax is similar for both languages. The syntactic ambiguity is preserved in the output text in the same way as the input text. Therefore, trying to resolve syntactic ambiguities is useless for our problem.

On the other hand, ambiguities involved in morphological and semantic analyses should be resolved for translation from Turkish to Azeri. Words that can be morphologically analyzed in more than one way due to several reasons are morphologically ambiguous. We have examined the characteristics of these words and categorized them according to reasons causing the ambiguity.

In the case of semantic analysis, words with multiple word senses cause the ambiguity. Some root words do not convey a single meaning. They can be used for as many different meanings as the number of their word senses. Therefore in order to find the correct equivalent of a Turkish word in Azeri, the word sense ambiguity should be resolved.

Another phenomena causing ambiguity in semantic analysis is *pronominal anaphora* [18]. In written discourse people may use certain instruments for pointing back in the discourse context to individuals, objects, events, times and concepts mentioned previously. The use of such a pointing device is called anaphora. Short constituents such as pronouns and definite noun phrases referring to more detailed descriptions elsewhere in the text are called anaphors. As an example consider the Turkish sentence, *Ali Ayşe'ye arabasını sordu.* It can be interpreted in two ways: *Ali asked Ayşe about Ali's(his) car.* or *Ali asked Ayşe about Ayşe's(her) car.* In order to identify the correct meaning expressed by the sentence, the pronominal anaphora should be resolved with semantic information obtained from the surrounding text. However, the disambiguation is not necessary for translation from Turkish to Azeri. When the sentence is translated into Azeri, the resulting text preserves the same anaphoric relation. Therefore, resolution of pronominal anaphora can be ignored in our problem.

We then should deal with the ambiguities resulting from morphological and semantic analysis. Thus, we have classified ambiguous Turkish words into two major categories:

1. word sense ambiguous words
2. morphologically ambiguous words

The Turkish words and their characteristics in both categories are presented below.

4.4.1. Word Sense Ambiguity

Word sense ambiguity exists in Turkish as there are words with multiple word senses. So, in this case, the Turkish word can be morphologically parsed in just one way, but the root word has multiple word senses causing multiple equivalents in Azeri.

Linguists distinguish between *homonyms* and *polysemes* [1]; *homonyms* are words like *bank* which have two or more distinct and unrelated meanings (geological feature or financial institution); *polysemes* are words like *face* which reflect different shades of meaning according to context. They distinguish also between *homophones* (words which sound the same but have different meanings) such as *pear, pair,* and *pare*, and *homographs* (words which are spelled the same but have different meanings) such as *tear* (crying versus ripping). Fortunately, the homophone problem is irrelevant since MT deals only with written texts. For resolving the ambiguities it is also immaterial whether the source language word is homograph or polyseme, since in both cases the ambiguous word has multiple equivalents in the target language, and the problem is to select the correct equivalent for its current usage.

Sometimes the target language vocabulary makes finer sense distinctions than source language. For instance, the verb *know* may be conveyed by *wissen* or *kennen* in German; similarly the English word *river* may be either *Fluss* or *Strom* in German. In either case

English words do not have more than one meaning. Instead, German makes distinctions which English does not. Nevertheless, in the context of a MT system the problem of selecting the correct target language form is much the same as when source language form is a genuine homograph or polyseme. Therefore we refer to all of these words as word sense ambiguous and treat them in the same manner in order to resolve the ambiguity.

In this case, the problem is to find the correct equivalent of the root word in Azeri for its current usage. For instance, the Turkish word *alay(regiment / mockery)* has two word senses, and the word *alaya* has a single parse :

alay(regiment / mockery) - a(dative)

The word is ambiguous, since the root word itself is ambiguous. Thus, the word sense ambiguity has to be resolved in order to translate the Turkish word *alaya* into Azeri.

However, if the equivalent of a Turkish word is the same in Azeri for each word sense, there is no need to resolve this ambiguity in translation from Turkish to Azeri. For example, the Turkish word *dön-* has two word senses *to revolve* and *to return*, and its Azeri equivalent is *dön-* for both senses. So, trying to resolve the word sense ambiguity is unnecessary in this case. The Turkish words *hat(feature / line)*, *pazar(Sunday / bazaar)* and *uyuş-(to get along together / to get numb)* are some other examples of this kind of words.

The list of word sense ambiguous words in the "Comparative Dictionary of Turkish Dialects" [13] that should be disambiguated in the translation process is given in Appendix A.

4.4.2. Morphological Ambiguities

Morphological ambiguity is the major ambiguity category in machine translation involving agglutinative languages, and it results from words with multiple morphological parses. Different types of morphological ambiguities exist in Turkish resulting from its agglutinative structure. We have so far identified 5 types of them :

1. root words that are similar in form to stems with inflectional suffixes
2. root words that are similar in form to stems with derivational suffixes
3. suffixes with multiple surface forms
4. similar suffixes
5. identical suffixes

These cases are discussed below together with illustrative examples.

4.4.2.1. Root Words that are Similar in Form to Stems with Inflectional Suffixes

In this case, the Turkish word has multiple parses with different root words. For instance, the Turkish word *halka* has two morphological parses:

halka(ring)
(ring)
halk(people) - a(dative)
(to the people)

Another example of this kind is the Turkish word *ilim*, which has also two morphological parses:

ilim(science)

(science)

il(city) - im(1st singular possessive)

(my city)

So, the reason of this type of morphological ambiguity is the existence of a Turkish root word that is similar in form to another root word with inflectional suffixes.

The close relation between Turkish and Azeri allows us to ignore some of the ambiguous words of this kind in the translation process. For example, the Turkish word *eser* is an ambiguous word of this kind, and it has the following two morphological parses:

eser(work of art)

äsär

es(to blow) - er(aorist suffix)

äs - är

Since, the Azeri equivalents are the same for both parses, *eser* can be translated into Azeri as *äsär* without any disambiguation process. The Turkish words *altmış/alt, arka/ark, kızar-/kız-, oku-/ok, yarasa/yara-* are also examples of these words as illustrated in Table 4.3.

Table 4.3. Examples of ambiguous words that can be ignored in translation from Turkish to Azeri

Turkish word	Azeri equivalent
altmış(sixty) alt(bottom) - mış(past indefinite tense)	altmış alt - mış
arka(the back) ark(canal) - a(dative)	arxa arx - a
kızar(to turn red) kız(to be angry) - ar(aorist)	gızar gız - ar
oku(to read) ok(arrow) - u(accusative/2nd singular possessive)	oxu ox - u
yarasa(bat) yara(to be useful) - sa(desiderative)	yarasa yara - sa

We have determined the ambiguous Turkish root words of this kind by parsing each Turkish word with the Turkish morphological parser developed at Boğaziçi University [8]. Those with multiple morphological parses are identified as ambiguous. We have categorized these words according to the suffixes causing the ambiguity, in order to resolve the ambiguities effectively. These suffixes are the following:

-(y)I (accusative)

-(s)I (3rd singular possessive)

-(y)A (dative, optative modal)

-DA (locative)

-(I)n (2nd singular possessive, 2nd single person for imperative modal)

-(I)m (1st singular possessive)

-(y)Iz (1st plural person)

-DI (past definite tense)

-mIş (past indefinite tense)

-Ar (aorist suffix)

-mA (negation suffix, nominating participle)

-lA (clitic)

-sA (desiderative modal, compound conditional tense)

-mAk (infinitive suffix)

Ambiguous Turkish words of this kind that should be disambiguated in translation from Turkish to Azeri are given in Section B.1 in Appendix B according to the above categorization.

4.4.2.2. Root Words that are Similar in Form to Stems with Derivational Suffixes

There exist Turkish root words that are similar in form to other root words with derivational suffixes, and they naturally convey absolutely different meanings. For example, the word *kayış* has two morphological parses:

kayış(belt)
(belt)
kay(to slide) - ış(derivational suffix from verb to noun)
(sliding)

Even though the word *kayış* is similar in form to word *kay-ış* the meaning *belt* has no relation with *sliding*. The word *kayış* may be assumed to be word sense ambiguous, since it may be thought of as having two word senses, belt and sliding. However this is not the case in the "Comparative Dictionary of Turkish Dialects" [13]. Namely, the meaning *sliding* is not given as a word sense of the word *kayış*. Therefore, we prefer to accept the words *kayış* and *kay-ış* as two different words, and classify them as ambiguous due to the reason introduced in this section.

The close relation between Turkish and Azeri also allows us to ignore some of the ambiguous words of this kind in the translation process. The Turkish word *dolan* is an

example of these words. It has two morphological parses:

dolan(to wander around)
dolan
dol(to fill up) - an(derivational suffix from verb to adjective)
dol - an

Since, the Azeri equivalents are the same for both parses, *dolan* can be translated into Azeri as *dolan* without any disambiguation process. The Turkish words *kayık/kay-, kırış-/kır-, geliş-/gel-* are other examples of these words as illustrated in Table 4.4.

Table 4.4. Examples of ambiguous words that can be ignored in translation from Turkish to Azeri

Turkish word	Azeri equivalent
kayık(boat) kay(to slide) - ık(derivational suffix from verb to adjective)	gayıg gay - ıg
kırış(to become wrinkled) kır(to break) - ış(derivational suffix from verb to noun)	gırış gır - ış
geliş(to develop) gel(to come) - iş(derivational suffix from verb to noun)	gäliş gäl - iş

Ambiguous Turkish root words of this kind are already determined together with the ones presented in Section 4.4.2.1. We have identified them, and categorized them in the same manner. Suffixes causing this kind of ambiguity are the following:

-(I)k (derivational suffix from verb to adjective)
-(I)ş (derivational suffix from verb to noun)
-CA (derivational suffix from noun to adjective or adverb)
-(y)An (derivational suffix from verb to adjective)

The ambiguous Turkish words of this kind that should be disambiguated in translation from Turkish to Azeri are given in Section B.2 in Appendix B according to the above categorization.

4.4.2.3. Suffixes with Multiple Surface Forms

Some suffixes in Turkish have multiple surface forms depending on the surface structure of the words that they would be affixed. For instance, the dative suffix -(y)A has the forms -yA and -A, two of its allomorphs, depending on the last character of the word it would be affixed. It is affixed to the word *masa(table)* as *masa-ya*, whereas to the word *bisiklet(bicycle)* as *bisiklet-e*. This may cause morphological ambiguity for words like *ada(island)* and *aday(candidate)*, when the word *adaya* is to be translated into Azeri. It has two morphological parses :

ada(island) - ya(dative)
(to the island)
aday(candidate) - a(dative)
(to the candidate)

The suffixes affected by the morphophonemic rules of phoneme insertion and elision cause this kind of ambiguity. We have determined this kind of suffixes for each part of speech. For example, the following suffixes that can be affixed to nouns are of this kind.

derivational suffixes:
-(A)l, -(i)k, -(i)st, -(i)zm

inflectional suffixes:
-(I)m, -(I)mIz, -(I)n, -(I)nIz, -(n)In, -(s)I
-(y)A, -(y)I, -(y)IlAn, -(y)InAn, -(y)lA

The letters that may be deleted due to morphophonemic rules are I, A, **n**, **y** and **s**. Therefore nouns that differ one letter in length are ambiguous, if the last letter of the longer one is among these letters and the suffix is affixed to the other one without any letter deletion, e.g. the words *elmas (diamond)* and *elma (apple)* in the word *elması (his/her apple / his/her diamond)*. So we identify ambiguous nouns by searching the root words lexicon for nouns satisfying this condition. The same process is repeated for each part of speech. The set of letters that might be deleted due to morphophonemic rules are I and A for adjectives, and I, A and **y** for verbs.

The ambiguous words of this kind that are caused by the deletion of the vowels I and A are also identified in the case of words that are similar in form to stems with inflectional suffixes. This is because an ambiguous word with length n ending with one of these vowels can be derived from its pair whose length is n-1 with either the accusative suffix -(y)I or the dative suffix -(y)A, e.g. the words *aşı (vaccination)* and *aş (cooked food)*.

The following is the list of the ambiguous words of this kind according to the letters causing the ambiguity, excluding those explained above:

(s) (the only example -(s)I)

elma	hala	hassa	kısa	küme
elmas	halas	hassas	kısas	kümes

(y) (like -(y)lA)

ada	kala	ne
aday	kalay	ney

4.4.2.4. Similar Suffixes

Some Turkish suffixes have similar surface forms, and this may cause morphological ambiguities when they are affixed to root words with similar surface forms. For instance, the suffix -DAn(ablative) and -(y)An(derivational suffix from verb to adjective) have similar surface forms, and they cause morphological ambiguity in the word *devreden*:

devre (period / circuit) - den(ablative)
(from the period / circuit)
dövrä - dän
devret(transfer) - en(derivational suffix from verb to adjective)
(the one who is transferring)
tähvil ver - än

For some of the ambiguous Turkish words of this kind, there is no need to resolve the ambiguity in order to achieve the translation, since the corresponding Azeri words and the suffixes are also identical. For example, the Turkish word *gemin* has three morphological parses:

gem(bit) - in(2nd singular possessive)
(your bit)
gäm - in
gem(bit) - in(genitive)
(of the bit)
gäm - in
gemi(ship) - n(2nd singular possessive)
(your ship)
gämi - n

However, the Azeri equivalents of all parses are the same. So, the ambiguity is ignored in the translation process.

Each word category has a fixed set of suffixes that can be affixed to the words in it. We have determined these sets of suffixes for each word category. Then using this information we have affixed all possible suffixes to each Turkish word and morphologically parsed them. In this way the ambiguous Turkish words of this kind are determined.

In this exclusive search, since we have also used the suffixes with multiple surface forms, the words determined as ambiguous due to these suffixes are also found in this case. Moreover, some of the words determined as ambiguous due to the previously explained two other reasons are also found in this case, since they are also ambiguous due to this reason. For example consider the Turkish word *altın (gold)* and the suffix -(s)I (3rd singular possessive). When the suffix is affixed to the word, the resulting Turkish word has the following five parses:

altın (gold) - ı(3rd singular possessive)
(his/her gold)
altın (gold) - ı(accusative)
(the gold)
altı (six) - n(2nd singular possessive) - ı(accusative)
(your six)
alt (bottom) - ın(2nd singular possessive) - ı(accusative)
(your bottom)
alt (bottom) - ı(3rd singular possessive) - nı(accusative)
(his/her bottom)

Therefore the Turkish words *altın, altı* and *alt* are ambiguous words of this kind. However, they were also determined as ambiguous when the Turkish word *altın* is tested

in the case of words that are similar in form to stems with inflectional suffixes. This is because it can be morphologically parsed in four ways:

altın (gold)
altı (six) - n(2nd singular possessive)
alt (bottom) - ın(2nd singular possessive)
alt (bottom) - ın(genitive)

So, in this case the Turkish words with more than one morphological parse, excluding those that are already determined due to reasons explained in the above sections, are identified as ambiguous words of this kind. They are presented in Section B.3 in Appendix B.

4.4.2.5. Identical Suffixes

In this case the Turkish word has multiple morphological parses with a single root word. The reason of the ambiguity is the existence of identical suffixes. Identical Turkish suffixes can be classified into two groups:

1. Suffixes with multiple meanings
2. Suffixes that become identical due to different surface forms

Some Turkish suffixes have more than one meaning. For instance, the suffix -(I)m could be used as both 1st singular possessive and 1st single person, as in the following example:

benim(my) güzel(beauty) - im(1st singular possessive)
(my beauty)
ben(I) güzel(beautiful) - im(1st single person)
(I am beautiful)

The Azeri equivalent of the Turkish word *güzelim* in the first utterance is *gözälim*, whereas in the second one *gözäläm*. Therefore the ambiguity should be resolved to find the correct translation.

Some other Turkish suffixes become identical depending on words with different surface forms. For instance the suffixes -(y)I(accusative) and -(s)I(3rd singular possessive) become -I when they are affixed to a word ending with a consonant. For instance, the word *bisikleti* is ambiguous :

bisiklet(bike) - i(3rd singular possessive)
(his/her bike)
bisiklet(bike) - i(accusative)
(the bike)

Since the Azeri equivalent of the word *bisiklet* is *velosiped*, which also ends with a consonant, there is no need to resolve the ambiguity. However, this is not valid for the Turkish word *erik(plum)*, as *eriği* has two morphological parses :

erik(plum) - i(3rd singular possessive)
(his/her plum)
erik(plum) - i(accusative)
(the plum)

Since the Azeri equivalent of *erik* is *alça*, the ambiguity should be resolved to decide whether to translate the word *eriği* as *alçanı* or *alçası*. So, for these two suffixes the morphological ambiguity should be resolved if the Azeri equivalent of the Turkish word ends with a vowel.

The ambiguous identical suffixes are determined in the following way. First, we form a list of generic words that includes all the morphophonemic alternations to be used as a test set. The words in this set are the following:

e	a	i	ı	u	ü	o	ö
ed	ad	id	ıd	ud	üd	od	öd
et	at	it	ıt	ut	üt	ot	öt

Next, for each root word category we determine the word categories a root word in this category can reach with free jumps in the transition network representation of the morphotactic rules. Then for each word category in the transition network, we identify the root word categories from which these categories can be reached with free jumps. Finally, the identical suffixes were found using the following algorithm given in Figure 4.2.

```
for all word categories in the transition network do
{the one under translation is shown as category(i)}
  for all suffixes that can be affixed to words in the category(i) do
  {the suffix under consideration is shown as suffix(i)(j)}
    for all possible root word categories from which the category(i) can be reached with free jumps do
    {the root word category under consideration is shown as rcategory(k)}
      for all artificial test words do
      {tword(m)}
        form the word to be tested as tword(m) + suffix(i)(j) and its category is rcategory(k)
        find morphological parses of the word
        if number of parses > 1 then
          store the related information in the ambiguity file
      end for
    end for
  end for
end for
```

Figure 4.2. Algorithm for Finding Identical Suffixes

After manual postprocessing the parses obtained from the above algorithm, the ambiguous identical suffixes are found. Postprocessing involves the elimination of the redundant parses. The parses produced by the identical suffixes are presented in Appendix C.

Tracing the algorithm above for the word category v2, which is a verb category, may help to illustrate this complicated process. The only suffix that can be affixed to the words in the category v2 is -Ir, and it can be reached by free jumps from the root word categories v2 and v81. Therefore the following words are formed and morphologically parsed in order to find the ambiguous identical suffixes with -Ir.

eir	edir	etir
aır	adır	atır
ır	ıdır	ıtır
iir	idir	itir
our	odur	otur
öür	ödür	ötür
uur	udur	utur
üür	üdür	ütür

The categories of these words are v2 and v81. At the end of the testing the following parse is found for the words ıdır, ıtır, idir, itir, odur, otur, ödür, ötür, udur, utur, üdür, ütür.

v2 - v3 - ır
v2 - v16 - v18 - v73 - v19 - v24 - v25 - n0 ır
v2 - v16 - v18 - v73 - v19 - v24 - v25 - v42 ır

All of them except the first one are eliminated in the postprocessing, since they are the redundant copies having the same information. The one maintained can be seen in

Appendix C. In this way the ambiguous identical suffixes are found for the suffix -Ir in the word category v2.

By using the algorithm presented in this section only two suffixes or a suffix with a suffix sequence can be identified as identical. However, a suffix sequence can be identical with another suffix sequence as well. For instance, the suffix sequence -(s)I (3rd singular possessive) and -(n)I (accusative) become identical with the suffix sequence -(I)n (2nd singular possessive) and -(y)I (accusative), when both are affixed to a word ending with a consonant. The Turkish word *defterini* which has the following two morphological parses illustrates this fact.

defter(notebook) - i(3rd singular possessive) - ni(accusative)
defter(notebook) - in(2nd singular possessive) - i(accusative)

The algorithm presented in this section can also be used to determine the identical suffix sequences by allowing the affixation of two or more suffixes. Although we did not carry out this process, we manually find out some obvious identical suffix sequences. The following is the list of these suffix sequences:

-(s)I -(n)A
-(I)n -(y)A

-(s)I -(n)I
-(I)n -(y)I

-(s)I -(n)In
-(I)n -(n)In

-(s)I -(n)dA
-(I)n -DA

-(s)I -(n)dAn
-(I)n -DAn

4.5. Methods Used in Resolving Ambiguities

Ambiguities can be resolved using the semantic information obtained from the text. Different kinds of semantic information at various levels may be used to resolve the ambiguities in translation. This depends on what is used as semantic information and in what structure. There are different types of semantic information and structures to represent them.

We utilize semantic features similar to those presented by Stoop [4], Boguraev et al. [19] and Sowa [20], the concept structure similar to those presented by Tomabechi [21] and Nogier et al. [22], and the collocation info structure similar to those presented by Nirenburg et al. [2] and Boguraev et al. [19] in order to resolve ambiguities. These structures are discussed below.

4.5.1. Semantic Features

Semantic features convey semantic information about words. Semantic features used for nouns, pronouns and proper nouns are displayed in Table 4.5.

Table 4.5. Semantic features for nouns, pronouns and proper nouns

part of speech	Semantic feature
noun	animate, inanimate, human, animal, plant, concrete/abstract, countable/uncountable
pronoun	human
proper noun	human, nation, country, city(town, village, region), sea(ocean, lake, river), mountain, language

We also use subcategories of the parts of speech as semantic features. The subcategories for each part of speech is displayed in Table 4.6.

Table 4.6. Subcategories of parts of speech

part of speech	Subcategories
pronoun	reflexive pronoun, personal pronoun, demonstrative pronoun, indefinite pronoun, interrogative pronoun, pronominal pronoun
adjective	qualificative adjective, demonstrative adjective, interrogative adjective, numeral adjective, indefinite adjective, time adjective
adverb	time adverb, place adverb, quantity adverb, quality adverb, interrogative adverb, demonstrative adverb
verb	transitive, intransitive, dative

Moreover, we define some other semantic features specific to a small range of words. The full list of these features is given in Appendix D and the following are the examples excerpted from this list:

time unit
weapon
object that can be bound
object that can be driven
organ of a human or an animal

All the semantic features presented in this section are utilized in collocation info structure to resolve ambiguities.

We do not intend to define semantic features to represent the world knowledge in a specific domain. Instead, in order to resolve an ambiguity if we need to identify whether a word has a certain property or not, we define this property as a semantic feature. For example, the Turkish word *cilt* has two word senses *skin* and *binding*. In order to resolve

this word sense ambiguity, we need to know whether an object can have a binding or not. Therefore, we have defined a semantic feature *object that can be bound* and identified all the words having this feature. So the ambiguity is tried to be resolved using this information.

4.5.2. Concept Structure

Concept structure is made up of the so-called concepts. To utilize the concept structure for an ambiguous word, a concept is defined and associated with it. In this way, the word becomes dependent on this concept. If an ambiguous word is dependent on a concept it means that the utterance under translation denotes this word if the related concept is active. In order for a concept to be active, at least one of the words that are related to it should have been used in the currently analyzed text. Therefore, together with each word, the numeric codes of the concepts it can activate are stored. The concepts that we have defined are as follows:

military, medicine, agriculture, device, electricity, mine,
grammar, navigation, mathematics, government, science, house,
space, weapon, religion, education, law, file, company, shop,
press, apartment house, architecture

We do not intend to define concepts to represent the complete world knowledge. Instead, we define concepts and associate the related words with them just to resolve the ambiguities. In this sense, our work is different from Schank's *Conceptual Dependency Notation(CD)* [23], which was devised to serve as semantic representation of the world knowledge. CD has its vocabulary for its specific domain, namely the sets of predicates, functions, and constants. However, we do not employ such a vocabulary, since our aim was not to represent the world knowledge.

Before translating a sentence, all the words in the sentence are allowed to activate their related concepts. By means of this, one sentence lookahead is performed in concept structure.

Concept 0 is utilized to indicate that if none of the other alternatives is identified as the correct meaning denoted by the utterance under translation, the ambiguous word associated with concept 0 is valid.

A concept is assumed passive if it is not activated. Once a concept is activated, it stays active during the analysis of the whole text. A concept may be sentence-active or text-active depending on whether or not the word activating it is used in the currently analyzed sentence. The priority of a sentence-active concept is higher than the text-active one.

For example, the Turkish word *uydu* is ambiguous due to root words that are similar in form to stems with inflectional suffixes. It has two morphological parses:

uydu(satellite)
uy(to fit) - du(past definite tense)

We have defined *space* as a concept, and the word *uydu* is associated with this concept. Consider the Turkish sentence *Uydu yörüngede dönüyor.* It has the following morphological parses:

uydu(satellite) yörünge(orbit) - de(locative) dön(to revolve) - üyor(present tense)
uy(to fit) - du(past definite tense) yörünge(orbit) - de(locative) dön(to revolve) - üyor(present tense)

The word *uydu* is ambiguous since it has two equivalents in Azeri as *peyk(satellite)* and *uydu(fitted)*. For this case, since the word *yörünge* has activated the concept *space*, *peyk*

is selected as the equivalent Azeri word as it is dependent on the concept *space*. So the sentence is translated into Azeri as

Peyk orbitada dönür.

4.5.3. Collocation Info Structure

In the collocation info structure, co-occurrence information of words is used to resolve ambiguities. To utilize the collocation info structure for an ambiguous word, those words that are used together with it are stored as collocations of the word in the syntactic forms it might appear in Turkish. A special notation is employed to express these syntactic structures. The symbols used in this notation are presented in Table 4.7 and the generic suffix names that we introduce for referring to the suffix meanings are presented in Table 4.8. This notation is also used in the examples presented in the following sections. The utterance under translation denotes this word if one of the collocations of the word is true, which means that the word is found to be used with its collocation in the correct syntactic form.

Other than the co-occurrence information of specific words, some other generic co-occurrences of words are also utilized in the collocation info structure. These are the following:

1. Simple syntactic rules of Turkish, e.g. an adjective precedes a noun and an adverb precedes a verb.

2. The subcategory information of verbs as transitive, and intransitive, and in the case of transitive verbs dative objects, e.g. *problem-i düşün- (to think about the problem), yalancı-ya kan- (to be persuaded by the liar).*

3. The possessive, genitive and case agreements in noun phrases, e.g. *senin kitab-ın (your book), bu at-a (to this horse).*

Collocations valid for more than one ambiguous word are defined as macro collocations. In this way, words with these collocations just refer to macro collocation definitions instead of explicitly defining them. This structure is mostly utilized for the ambiguous words that are similar in form to stems with inflectional or derivational suffixes. This is because the ambiguous words caused by the same suffix mostly have same collocations resulting from the nature of the suffix.

Collocations of the alternative meanings of an utterance should be specific to each meaning. This means that the collocations that can be used with more than one alternative ambiguous word can not be stored as collocations of these alternatives, since they can not be utilized to resolve the ambiguity. Therefore the collocation information for alternative ambiguous words should be mutually exclusive.

In locating the correct equivalent of a root word in Azeri, the hints obtained from collocation info structure are given a priority higher than those obtained from concept structure. One sentence lookahead is performed in the collocation info structure as well.

Table 4.7. Symbols used in collocation info definitions

collocation info definition element	symbol
word with the part of speech	ca(a..z)
word with the specific semantic feature	c(0..99)
word with the noun semantic feature	0..9
word with the proper noun semantic feature	o(0..9)
word with the adjective subcategory	a(0..9)
word with the adverb subcategory	d(0..9)
word with the pronoun subcategory	p(0..9)
word with the verb subcategory	v(0..9)
macro collocation info definition	co(0..99)
the ambiguous word or the suffix (constant suffixes may be affixed to this word)	-
a generic suffix name follows (the word should have the suffix) (if no suffix name follows, the word should have at least one suffix)	+
no suffixes can be affixed to the word	(n)
alternative collocations	(,)
sentence with the mood (int for interrogative, exc for exclamation and sta for statement)	mood( )
possible(not restricted) collocation	[ ]

Table 4.8. Generic suffix names

Suffix	Generic Name
locative (-DA)	loca
ablative (-DAn)	abla
genitive (-(n)In)	geni
dative (-(y)A)	dati
accusative (-(y)I)	accu
any of the case endings	case
pronominal (-ki)	pron
negation (-mA / -mAz)	negp
potential (-Abil)	cyet
aorist (-Ir / -Ar)	tlte
past definite tense (-mIş)	pate
present tense (-mAktA)	ptte
necessitative modal (-mAlI)	nete
past indefinite tense (-DI)	pite
desiderative modal (-sA)	dete
present tense (-Iyor)	prte
optative modal (-(y)A)	opte
future tense (-(y)AcAk)	fute
imperative modal (-)	imte
interrogative particle (-mI)	intp
compound imperfect tense (-DI)	eech
compound narrative tense (-mIş)	eecr
compound conditional tense (-sA)	eecs
predicative (-DIr)	eecg
clitic (-(y)lA)	iles
infinitive (-mAk)	infs
nominating particle (-mA)	noms
plural (-lAr)	plrp
x.th singular/plural possessive	psxs / psxp
x.th singular/plural personal ending	pexs / pexp
any of the possessive suffixes / personal endings	ps / pe
x.th derivational suffix from word category a to b	dsabx

For example, the Turkish word *uydu* is ambiguous as explained in the previous section. This time consider the Turkish sentence *Bu anahtar kapıya uydu.* It can be morphologically parsed in two ways:

bu(this) anahtar(key) kapı(door) - ya(dative) uydu(satellite)
bu(this) anahtar(key) kapı(door) - ya(dative) uy(to fit) - du(past definite tense)

So, the word *uydu* is ambiguous, and its Azeri equivalents are *peyk(satellite)* and *uydu(fitted)*. That means the ambiguity should be resolved. This time the concept *space* is not active, whereas one of the collocations of *uy-* is true. *uy-* has the following 3 collocations:

-an ca(n)
ca(n)+dati [ca(d)] -
ca(a)+dati -

The second collocation *ca(n)+dati [ca(d)] -* is true for this sentence as *ca(n)(kapı)+dati(ya) -(uydu)*. Therefore the correct morphological parse of the word is decided as *uy(to fit) - du(past definite tense)*. So the Turkish sentence is translated into Azeri as:

Bu açar gapıya uydu.

4.6. Resolving Ambiguities

This section explains resolving each kind of ambiguity using the methods explained above.

4.6.1. Word Sense Ambiguity

To translate the word sense ambiguous words into Azeri, the ambiguity should be resolved. We employ the concept structure and the collocation info structure to resolve this kind of ambiguity. For each sense of the word sense ambiguous word either the concept structure or the collocation info structure is used to resolve the ambiguity.

As an example, consider the following Turkish sentence and its output obtained from the morphological parser:

Komutan bu alaya tayin edildi.
(The commander was appointed to this regiment)
Komutan(The commander) bu(this) alay(regiment)-a(dative)
tayin et(to appoint)-il(derivational suffix from verb to verb)-di(past definite tense)

The words *komutan* and *tayin edildi* are translated as *komandir* and *tä'yin edildi*, respectively. However the word *alay* is ambiguous since it has two equivalents in Azeri as *alay(regiment)* and *lağ(mockery)*. For this case, since *regiment* word sense of the Turkish word *alay* is dependent on concept *army* and the word *komutan* has activated the concept *army, alay* is selected as the equivalent Azeri word. So the sentence is translated into Azeri as:

Komandir bu alaya tä'yin edildi.

As a second example consider the Turkish sentence

Komutan askerle alay etti.
(The commander mocked the soldier)

The words *komutan* and *asker* are translated as *komandir* and *äsgär*, respectively.

However the word *alay* is again ambiguous. In this case, although the word *komutan* is activated the concept *army*, *alay* is translated as *lağ* since the following collocation of the word sense *mockery* is true

ca(n)+iles -(n) (etmek,geçmek)

as *ca(n)(asker)+iles(le) -(n)(alay) etmek.* So the sentence is translated into Azeri as:

Komandir äsgärlä lağ etdi.

This example also illustrates the fact that collocation info structure has a higher priority in comparison with concept structure.

Some of the word sense ambiguities can not be disambiguated by using either the concept structure or the collocation info structure. As an example of this kind consider the Turkish sentence *Alaydan hiç hoşlanmaz.(He/she does not like the regiment/mockery at all).* The morphological parse of this sentence is:

Alay(regiment/mockery) - dan(ablative) hiç(at all) hoşlan(to like) - maz(negation)

The words *hiç* and *hoşlanmaz* are translated as *heç* and *xoşlanmaz*, respectively. However the word *alay* is ambiguous since it has two equivalents in Azeri *alay(regiment)* and *lağ(mockery)*. In this case, since neither the concept *army* is active, nor any of the collocations of regiment is used with it, the ambiguity can not be resolved. Therefore the sentence is translated into Azeri as:

(Alay / lağ) heç xoşlanmaz.

4.6.2. Morphological Ambiguities

This kind of ambiguities can be divided into two groups according to their treatment in the disambiguation process:

1. Ambiguities involving multiple root words
2. Ambiguities involving identical suffixes

4.6.2.1 Multiple Root Words

All the morphologically ambiguous Turkish words due to reasons other than identical suffixes have multiple morphological parses with different root words. Therefore for all the cases, the correct root word should be decided for the current usage of the Turkish word in order to resolve the ambiguity.

Collocation info structure and concept structure is also utilized for these ambiguities. The related collocation info or concept structure information is collected and assigned to each root word of this kind.

As an example, consider the Turkish sentence *Pastan battım.* It can be morphologically parsed in two ways:

pas(rust) - tan(ablative) bat(to soil) - tı(past definite tense) - m(1st single person)
pasta(cake) - n(2nd singular possessive) bat(to soil) - tı(past definite tense) - m(1st single person)

Since the Azeri equivalents of all the word senses of the Turkish word *bat-* are the same, this word sense ambiguity is ignored in the translation. But, the words *pasta* and *pas* are ambiguous due to similar suffixes, and their Azeri equivalents are *tort* and *pas*,

respectively. This means that the ambiguity should be resolved. Collocation info structure is used for both words. The following collocation of the word *pas* is true for this case

-tan (batmak,geçilmez olmak)+(pe1s,pe1p,pe2s,pe2p,pe3p)

as *-tan bat+pe1s*. Therefore the correct morphological parse of the word is decided as *pas(rust) - tan(ablative)*. So the Turkish sentence is translated into Azeri as:

Pasdan batdım.

As another example, consider the Turkish sentence *Yetmiş tane elbise sattık*. It can be morphologically parsed in two ways:

yetmiş(seventy) tane(piece) elbise(dress) sat(to sell) - tı(past definite tense) - k(1st plural person)
yet(to suffice) - miş(past indefinite tense) tane(piece) elbise(dress) sat(to sell) - tı(past definite tense) - k(1st plural person)

The words *yetmiş* and *yet-* are ambiguous due to words that are similar in form to stems with inflectional suffixes. Since their Azeri equivalents are *yetmiş* and *yetiş-*, respectively, the ambiguity should be resolved. Collocation info structure is utilized for both words. The following collocation of the word *yetmiş* is true for this case

- (tane, adet)

as *- tane*. Therefore the correct morphological parse of the word is *yetmiş(seventy)*, and the Turkish sentence is translated into Azeri as:

Yetmiş dänä paltar satdıg.

4.6.2.2 Identical Suffixes

The ambiguous Turkish words in this category have multiple morphological parses due to identical suffixes. The root words for all the possible parses are the same. So the problem is to decide among the alternative suffixes.

Collocation info structure is used for these ambiguities. Collocation information is collected for the words derived with the ambiguous suffixes or suffix sequences. This information is used to resolve the ambiguities resulting from the words affixed with these ambiguous suffixes.

As an example, consider the Turkish sentence *Bu senin arabanın kapısı.* It has the following two morphological parses:

bu(this) sen(you) - in(2nd singular possessive/genitive) araba(car) - nın(genitive) kapı(door) - sı(3rd singular possessive)
bu(this) sen(you) - in(2nd singular possessive/genitive) araba(car) - n(2nd singular possessive) - ın(genitive) kapı(door) - sı(3rd singular possessive)

The word *senin* and *arabanın* are morphologically ambiguous due to identical suffixes. The suffix genitive is true for the word *senin*, since the collocation - *c(n)+ps* is true for it as - *c(n)(araba)+ps(n)*. For the word *arabanın*, the following collocation of the suffix sequence -(I)n (2nd singular possessive), -(n)In (genitive)

senin [ca(a)] -

is true as *senin* -. Therefore the Turkish sentence is translated into Azeri as:

Bu sänin arabanın gapısı.

5. A TRANSLATOR FROM TURKISH TO THE AZERI LANGUAGE

This chapter explains the translator from Turkish to Azeri which is designed and implemented in Pascal in this study. The implementation is presented in terms of the contents and the structures of the lexicons and the algorithms.

5.1. Overview of the Translator

Translation is performed sentence by sentence. The translator first reads a sentence, and calls the Turkish morphological parser to realize the morphological analysis phase. In this way, it obtains all possible parses for all words of the sentence. Translation starts after all of the words in the sentence are morphologically parsed, since the information obtained from the morphological analysis is likely to be used in ambiguity resolution process. The overall structure of the translator is illustrated in Figure 5.1.

For each word one of the following three cases is true depending on the result obtained from the morphological parser:

1. no morphological parse
2. single morphological parse
3. multiple morphological parses

The translator assumes that the text is free of spelling errors. Therefore, before running the translator, the input text should be checked with a spell checker. We are using the one developed in Boğaziçi University [8]. So, during the translation if a word can not be morphologically parsed in any way, it means that the word does not exist in the translator's lexicons. In this case the Turkish word is given as the result of the translation together with the message "No Information".

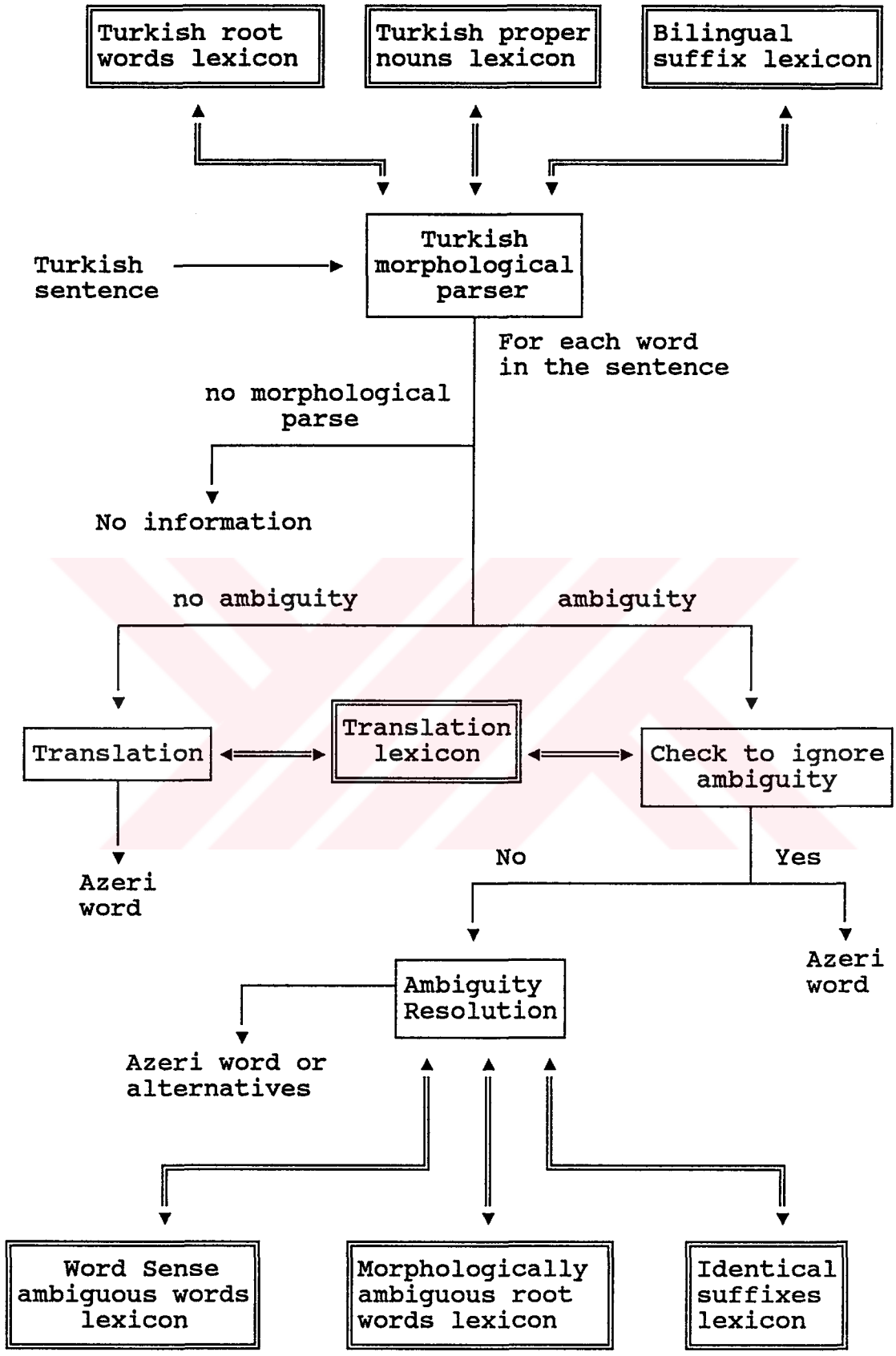

Figure 5.1. Overall Structure of the Translator

In case of a single morphological parse, if the word does not involve word sense ambiguity, it is translated into Azeri by replacing the Turkish root word and the suffixes with their Azeri equivalents. Proper noun root words are translated into Azeri without any modification. If it involves word sense ambiguity, it is tried to be disambiguated in the Resolve Word Sense Ambiguity routine by utilizing the concept and collocation info structures.

In case of multiple parses, translator first checks whether the ambiguity can be ignored or not. The check is performed by finding the Azeri equivalents of each parse and then comparing them. If the Azeri equivalents for each parse of the Turkish word are the same, then the ambiguity is simply ignored. However, it should be noted that if any one of the Turkish root words is word sense ambiguous, the ambiguity can not be ignored, since the Turkish word has alternative equivalents in Azeri at least for this parse.

If the word has more than one morphological parse, it is identified as ambiguous either due to multiple root words or due to identical suffixes. Resolve Multiple Root Words Ambiguity routine, which employs the concept and collocation info structures, is called to resolve the multiple root words ambiguity and Resolve Identical Suffix Ambiguity routine, which utilizes only the collocation info structure, is called to resolve the identical suffix ambiguity.

In case of multiple root words ambiguity, if any one of the root words is also word sense ambiguous, then Resolve Word Sense Ambiguity routine is also utilized to resolve the ambiguity. In case of identical suffix ambiguity, even if the ambiguity is resolved, the root word itself may be word sense ambiguous. If this is the case Resolve Word Sense Ambiguity routine is utilized to resolve the ambiguity.

Moreover, there exist some words which involve multiple root words and identical suffix ambiguity at the same time, e.g. the Turkish word *kalemi* as *kalem(pen)* + -(y)I (accusative), *kalem(pen)* + -(s)I(3rd singular possessive) or *kale(castle)* + -(I)m (1st

singular possessive) + -(y)I(accusative). If this is the case, since the Turkish word involves identical suffix ambiguity, it is treated as ambiguous due to this reason and Resolve Identical Suffix Ambiguity routine is utilized to resolve the ambiguity. If the ambiguity can not be resolved by this routine, and it also involves the multiple root words ambiguity then Resolve Multiple Root Words Ambiguity routine is employed to resolve the ambiguity.

If the ambiguity is resolved the equivalent Azeri word, otherwise the alternative translations are given as the result of this word's translation.

Translator employs 8 lexicons in the translation process. The Turkish root words lexicon, the Turkish proper nouns lexicon, and the bilingual suffix lexicon are used in the morphological analysis of Turkish words. The translation lexicon is utilized to find the Azeri equivalents of Turkish words, and their specific semantic features. The morphologically ambiguous root words lexicon, the word sense ambiguous root words lexicon, the identical suffixes lexicon, and the macro collocation info definitions lexicon are used in resolving ambiguities.

5.2. Lexicons

A translator must employ complete lexicons and well-designed structures to effectively access them. We have employed the following 8 lexicons in the translator:

1. Turkish root words lexicon
2. translation lexicon
3. morphologically ambiguous root words lexicon
4. word sense ambiguous root words lexicon
5. identical suffixes lexicon
6. macro collocation info definitions lexicon
7. bilingual suffix lexicon
8. Turkish proper nouns lexicon

The contents and the structures of each lexicon are discussed below. The entries in each of these lexicons, except the suffix lexicon which has a considerably complex structure, are presented in Table 5.1. Since the translator is implemented using Turbo Pascal 6.0 compiler and it involves considerably large lexicons, it has not been possible to load the contents of these lexicons into main memory for fast access. Therefore all the lexicons except the suffix lexicon are kept in disk, and the information in them is obtained by file access.

Table 5.1. The content of the translator's lexicons

Lexicon	Number of entries	Information in each entry
Turkish root words lexicon	6900	1. Turkish word 2. its_part_of_speech 3. flags 4. semantic_features_for_nouns_and_pronouns 5. semantic_features_as_part_of_speech_ subcategories
Translation lexicon	6900	1. Azeri_equivalent 2. concept_activation_information_or_semantic_ features 3. index_of_the_morphologically_ambiguous_root_ words_lexicon_if_any
Morphologically ambiguous root words lexicon	595	1. Turkish_word 2. concept_structure_array 3. collocation_info_linked_list 4. wsflag 5. conceptf 6. colinfof
Word sense ambiguous root words lexicon	157	1. Turkish_word 2. wsno 3. array_of_word_sense_records 3.1. Azeri_equivalent 3.2. concept_structure_array 3.3. collocation_info_linked_list 3.4. conceptf 3.5. colinfof
Macro collocation info definitions lexicon	25	1. collocation_info_linked_list
Identical suffixes lexicon	49	1. suffix 2. collocation_info_linked_list
Turkish proper nouns lexicon	10000	1. Turkish_proper_noun 2. flags 3. semantic_features_for_proper_nouns

5.2.1. Turkish Root Words Lexicon

Part of the data in the Turkish root words lexicon necessary for the morphological analysis of Turkish, which includes the Turkish words, their parts of speech and a series of flags, is obtained from Department of Computer Engineering, Boğaziçi University [8]. If a certain flag is set for a word it means that either the word has the property represented by that flag or the suffix represented by that flag can be affixed to the word. In the current implementation 56 flags are used, 38 for nouns and adjectives, and 18 for verbs and adverbs. For a detailed description of these flags see "Internal Design Specification, A Spelling Checker and Corrector for Turkish" [24] .

Turkish words derived using derivational suffixes that do not exist in Azeri and those derived by the suffixes that are not applicable to some Azeri words, about 1000, and their related information are also added to the lexicon. Thus, the lexicon totally contains about 6900 words. If a Turkish word derived by a suffix that is applicable to some Azeri words is not stored in the lexicon, the suffix is assumed to be applicable to its root word. Moreover, it should be noted that if a Turkish word derived by the suffixes that are applicable to some Azeri words is added to the lexicon, the flags that allow the affixation of the suffixes deriving it should be set to 0.

The semantic features presented in Section 4.5.1 are added to all the words in the root lexicon. So, the structure of the lexicon is a file of records with the following definition:

```
rootrecord = record
        Turkish_word : string[maxwordlength];
        its_parts_of_speech : array [1..5] of categorytype;
        flags : array [1..38] of boolean;
        semantic_features_for_nouns_and_pronouns : array [1..10] of boolean;
        semantic_features_as_part_of_speech_subcategories : array [1..5] of subcategorytype;
end;
```

In this definition, maxwordlength denotes the maximum length of a Turkish word, categorytype is a user defined type to represent the word categories in the transition network, and subcategorytype is also a user defined type to represent the subcategories of the parts of speech. The same notation is used for all the record definitions in this chapter.

The current implementation uses an index of array type, each element of which is pointing to the first word in the group of words whose first two letters are the same. A representative listing of the lexicon including the first 150 entries is given in Section E.1 in Appendix E.

5.2.2. Translation Lexicon

The Azeri equivalents of Turkish words are obtained from "Comparative Dictionary of Turkish Dialects" [13]. About 400 of these words are not included in the root lexicon, since they can be derived from existing root words with derivational suffixes. The entries of this lexicon are in one-to-one correspondence with the entries of Turkish root words lexicon. Therefore, the Azeri words in this lexicon are the equivalents of the Turkish words located in the same entries in Turkish root words lexicon. So, this lexicon has also about 6900 entries. The Azeri equivalent for a word sense ambiguous Turkish word is stored as "ws" to indicate that it is word sense ambiguous. If an unambiguous Turkish word has multiple equivalents in Azeri only one of them is selected arbitrarily.

A flag (azerif) is employed to denote whether the equivalent of the Turkish word in Azeri is exactly the same as the Turkish word or not. By means of this flag, duplication of the Azeri equivalents of the Turkish words that are the same with the Turkish words, which are about 1100, is eliminated.

The Azeri equivalents of the Turkish words that are affected by the phonological variation rules, valid for all the Azeri words, are not stored in the lexicon, instead the same flag azerif is utilized. Therefore, while locating the Azeri equivalent of a Turkish word, if its Azeri equivalent is not explicitly stored in the lexicon, these phonological variation rules are automatically applied to the Turkish word.

Actually, we have attempted to utilize the phonological variation rules that are valid for some Azeri words in storing the Azeri equivalents of Turkish words. Our idea is the following. If the Azeri equivalent of a Turkish word is its phonological variation depending on certain phonological rules , instead of storing the Azeri word explicitly, only the flags representing these rules may be stored. However, since the number of Azeri words that are affected by these rules are quite low, this strategy seemed to be inefficient. Thus, it is not used in the current implementation.

As another attribute, concept activation information and specific semantic features are stored together with each word in an array containing 5 entries. These entries are either the numbers representing the concepts that the usage of the word activates or the numbers representing the specific semantic features that the word satisfies. The last attribute is the index of the corresponding entry in the morphologically ambiguous root words lexicon for the words that may cause morphological ambiguities.

So, the structure of the lexicon is file of records with the following fields:

```
translationrecord = record
        Azeri_equivalent : string[maxwordlength];
        Azerif : boolean;
        concept_activation_information_or_specific_semantic_features : array [1..5] of integer;
        Index_of_the_morphologically_ambiguous_root_words_lexicon_entry_if_any : integer;
end;
```

No index is employed for this lexicon, since the entries are in one-to-one correspondence with the entries of the Turkish root words lexicon, and they are not accessed without accessing to that lexicon. A representative listing of the lexicon including the Azeri equivalents of the Turkish words in Section E.1 of Appendix E is given in Section E.2.

5.2.3. Morphologically Ambiguous Root Words Lexicon

In the morphologically ambiguous root words lexicon an array to implement the concept structure and a linked list to implement the collocation info structure are stored for each entry corresponding to a morphologically ambiguous word in the translation lexicon. Instead of all the collocations of the ambiguous words, only the collocations specific to the resolution of the ambiguities are stored in the lexicon. We have identified only those morphologically ambiguous words that exist in our root words lexicon. The lexicon contains disambiguation information for 595 morphologically ambiguous words.

The concept array has 3 entries since in the current implementation a Turkish word is expected to depend on 3 different concepts at the same time. In each array entry, the numeric code of the concept on which the Turkish word is dependent is stored. The collocation info header pointer points to a linked list of nodes, each containing a collocation of the ambiguous Turkish word. If one of the collocations is used together with the word in the correct syntactic form, the corresponding Azeri word in the translation lexicon is chosen as the result of the translation. The semantic information in each entry is used in the disambiguation process involving the corresponding word in the translation lexicon.

A flag (wsflag in the declaration below) is employed to denote whether the Turkish word is also word sense ambiguous or not, and two additional flags are employed to speed up the process of checking whether the concept (conceptf) and collocation info structures (colinfof) are used in the disambiguation process of the word or not.

So, the structure of this lexicon is file of records with the following fields:

```
colinfopointer = ^colinforecord;
colinforecord = record
        collocation : string[maxcollocationlength];
        next : colinfopointer;
end;

morphambiguouswordrecord = record
        Turkish_word : string[maxwordlength];
        concept_structure_array : array [1..3] of integer;
        collocation_info_linked_list : colinfopointer;
        wsflag, conceptf, colinfof : boolean;
end;
```

Colinfopointer is a linked list structure to store the collocation definitions, and maxcollocationlength denotes the maximum length of a collocation definition. Since the entries of this lexicon are accessed via the translation lexicon, and the entry numbers are stored in it, no index is employed for this lexicon. The full listing of this lexicon is given in Section E.3 in Appendix E.

5.2.4. Word Sense Ambiguous Root Words Lexicon

Turkish words with multiple equivalents in Azeri, due to multiple word senses are stored in a separate lexicon in order to handle them effectively. We have identified only those word sense ambiguous words that exist in our root words lexicon. The lexicon contains 157 word sense ambiguous Turkish words, and disambiguation information for them.

An array implementing the concept structure, a linked list implementing the collocation info structure, the Azeri equivalent of the Turkish word, and two flags to represent

whether the concept and the collocation info structures are used in the disambiguation process of the word sense or not are stored for each word sense. The concept structure array and collocation info linked list are the same with the ones explained in Section 5.2.3. So, disambiguation information for each word sense is stored as a record with the following attributes:

```
wordsenserecord = record
        Azeri_equivalent : string[maxwordlength];
        concept_structure_array : array [1..3] of integer;
        collocation_info_linked_list : colinfopointer;
        conceptf, colinfof : boolean;
end;
```

An array of word sense records including these fields are stored together with each word and the number of its word senses. This array has 4 entries since the Turkish words in our root words lexicon have at most 4 word senses. So, the structure of this lexicon is file of records with the following fields:

```
wordsenseambiguouswordrecord = record
        Turkish_word : string[maxwordlength];
        array_of_word_sense_records : array [1..4] of wordsenserecord;
        wsno : integer;
end;
```

The current implementation uses an index of array type, each element of which is pointing to the first word in the group of words whose first two letters are the same. The full listing of the lexicon is given in Section E.4 in Appendix E.

5.2.5. Macro Collocation Info Definitions Lexicon

Collocations that are valid for more than one ambiguous word are defined as macro collocations. This lexicon contains these macro collocation definitions. In the current implementation we use 25 macro collocation definitions. So the structure of the lexicon is file of collocation info linked list. Since the macro collocation info definitions are referred with their entry numbers, no index is associated with this lexicon. The full listing of the lexicon is given in Section E.5 in Appendix E.

5.2.6. Identical Suffixes Lexicon

The identical suffixes lexicon is utilized in order to resolve the morphological ambiguities resulting from identical suffixes. A suffix may be identical with another suffix as well as with a suffix sequence. The lexicon currently contains 49 identical suffixes.

Generic names for each identical suffix is stored in an array of suffixes, because of the existence of identical suffix sequences. A pointer array used to implement the collocation info structure for each suffix or suffix sequence is also stored. So, the structure of the file is file of records with the following attributes:

```
identicalsuffixrecord = record
        suffix : array [1..3] of generic_suffix_symbols;
        collocation_info_linked_list : colinfopointer;
end;
```

Generic_suffix_symbols is a user defined type to represent the generic names of the suffixes. (see Section 4.5.3) The suffixes are stored in sorted order. The full listing of the lexicon is given in Section E.6 in Appendix E.

5.2.7. Bilingual Suffix Lexicon

Morphological parsing has special importance in translation between agglutinative languages. The key feature of morphological parsing is the suffix lexicon, which mirrors the transition network representation of the morphotactics of the language. The suffix lexicon employed in the morphological analysis of Turkish is also obtained from Boğaziçi University [8]. It is refined in order to be used in the translation process.

The morphophonemic and morphotactic rules of the Azeri are taken from the "Comparative Dictionary of Turkish Dialects" [13] and "The Language of Turks" [14]. Some of the information about these rules and derivational suffixes that does not exist in these references was extracted from the Azeri words and Azeri texts by analyzing them. Derivational suffixes that do not exist in Azeri were excluded from the suffix lexicon. The meanings of the suffixes as generic suffix names and their Azeri equivalents were added to it. A representative listing of the suffix lexicon, including the transitions for nouns and adjectives is given in Section E.7 in Appendix E. For a detailed discussion of this lexicon see "Internal Design Specification, A Spelling Checker and Corrector for Turkish" [24].

The data structure employed is a bucket structure. The bucket is an array of nodes, each belonging to a word state, which is the initial state of a transition. Each node in the bucket includes the name of the state, information on whether the words in this state are valid or not, and a pointer to the linked list of nodes holding information about the suffixes that can be affixed to the words in this state. Each node in the linked list contains the following information:

1. a Turkish suffix

2. its meaning as a generic suffix name

3. its Azeri equivalent

4. the final state of the word if this suffix is affixed to it

5. the flag number used to decide whether the suffix can be affixed to the word or not

6. the information whether the suffix is affixed to the end of the word or affixed separately

7. a pointer to the next node

The suffix lexicon is actually stored in a file and it is loaded into the bucket structure before the translator is run.

5.2.8. Turkish Proper Nouns Lexicon

The proper nouns lexicon was also obtained from Boğaziçi University [8]. It contains about 10000 Turkish proper nouns together with 13 flags for each that are used in their morphological analysis. For a detailed description of these flags see "Internal Design Specification, A Spelling Checker and Corrector for Turkish" [24]. The semantic features for the proper nouns are added to the lexicon. The structure of the proper nouns lexicon is file of records with the following fields:

```
propernounrecord = record
        Turkish_proper_noun : string[maxwordlength];
        flags : array [1..13] of boolean;
        semantic_features_for_proper_nouns : array [1..10] of boolean;
end;
```

The same index structure as the Turkish root words lexicon is utilized for this lexicon. A representative listing of the lexicon including the first 100 entries is given in Section E.8 in Appendix E.

5.3. Algorithms

The algorithms necessary to implement the translator are listed below. The main algorithm is followed by the others. Whenever appropriate and helpful for a better understanding, a brief trace of the algorithm is given with specific examples. The overall algorithm of the translator is illustrated in Figure 5.2. The symbol nop in the figure denotes the number of parses for a word. The list of the algorithms for the translator is as follows:

1. Main Algorithm
2. Modifying Translator Lexicons
3. Turkish Morphological Parser
4. Affixing Suffixes
5. Affixing a Suffix
6. No Ambiguity Case
7. Finding Collocation Index
8. Checking Collocation
9. Checking Macro Collocation Definition
10. Resolving Word Sense Ambiguities
11. Resolving Morphological Ambiguities
12. Resolving Identical Suffix Ambiguities

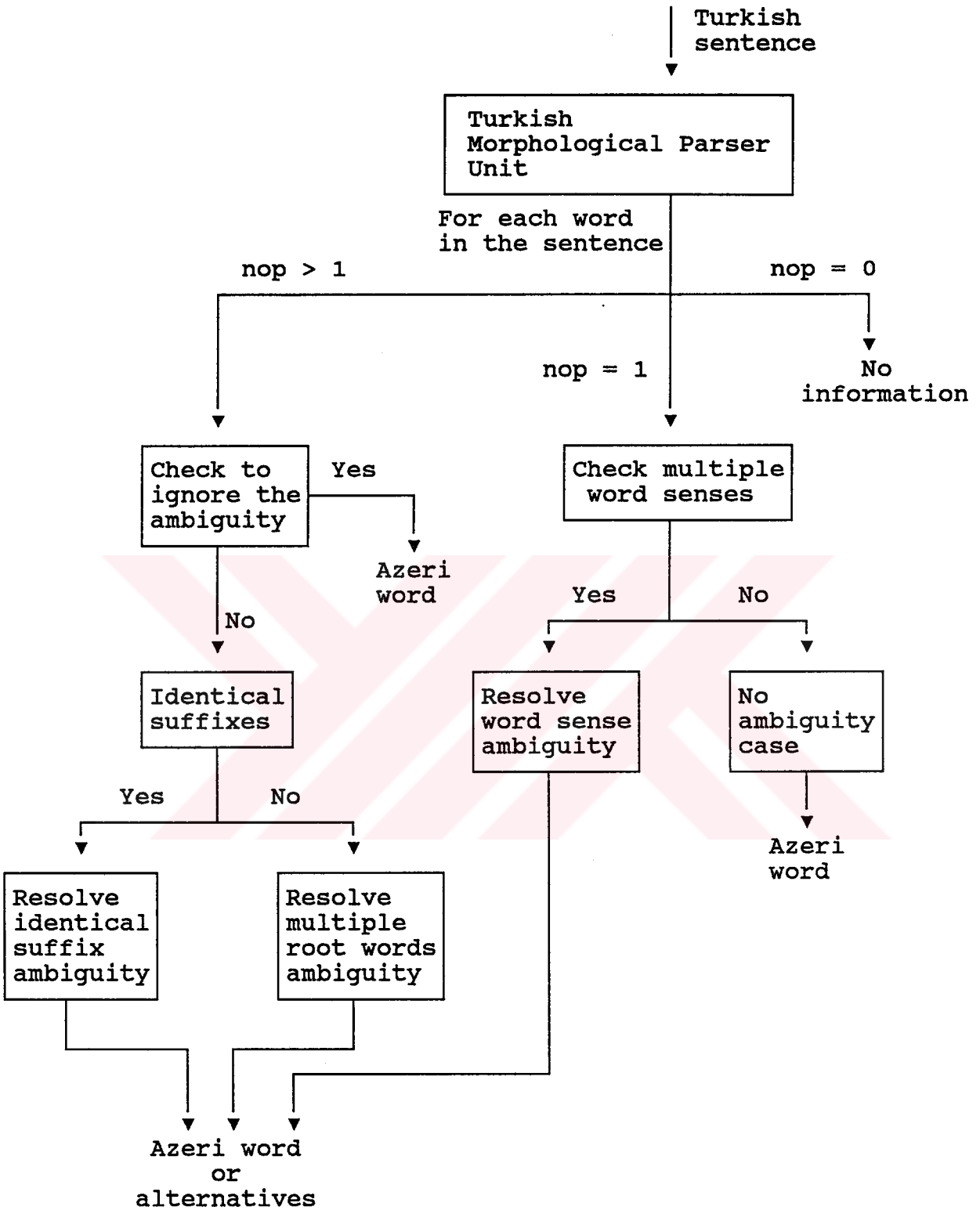

Figure 5.2. The Overall Algorithm of the Translator

5.3.1. Main Algorithm

The main algorithm initially checks the request for supervisor options. In case of such a request, if the user supplies the correct password, it allows the user to modify the translator's lexicons. The translator then loads the suffix lexicon into main memory and creates indexes for the Turkish root words, the Turkish proper nouns and the word sense ambiguous root words lexicons.

It begins the translation process by reading the Turkish text that will be translated into Azeri. Translation is performed sentence by sentence. First, it finds all the morphological parses of the words in the sentence by using Turkish morphological parser. Then depending on the result obtained form the parser, it determines whether the word involves any ambiguity or not, and the kind of the ambiguity if any. Finally, for each word of the sentence it calls appropriate routines to perform the translation. Figure 5.3 gives the algorithm in structured English.

```
ask the user for supervisor options
if gets a positive request then
 call Modifying Translator Lexicons routine
while not end of the Turkish text do
 read the sentence
 for each word in the sentence do
  call Turkish Morphological Parser unit and get the number of parses of the word and the related information
for each parse
  activate the concepts related with the word if any
 end for
{ translation }
for each word in the sentence do
  set okay to false
```

Figure 5.3. Main Algorithm

```
{ check to ignore ambiguity }
if number of parses is greater than one then
 form the Azeri equivalent for each parse
 if all of them are the same then
   if none of them is word sense ambiguous then
    output the translation as the Azeri equivalent of one of the parses since all are the same
    set okay to true
   end if
 end if
end if
if not okay then
 { No information }
 if number of parses is zero then
  perform no translation process and output the Turkish word itself with the prompt "No information"
 { nop = 1 and not a proper noun }
 else if number of parses is one and not a proper noun then
  if the root word is not word sense ambiguous then
   call No Ambiguity Case
   output the translation
  else
   call Resolve Word Sense Ambiguity
   if Resolved then
    output the translation
   else
    output the alternative translations
   end if
  end if
 { nop = 1 and proper noun }
 else if number of parses is one and a proper noun then
  call No Ambiguity Case
  output the translation
```

Figure 5.3. Main Algorithm (continued)

```
{ multiple root words ambiguity }
else if number of parses is greater than one and no identical suffix ambiguity then
 call Resolve Multiple Root Words Ambiguity
 if Resolved then
  output the translation
 else
  output the alternative translations
 end if
{ identical suffix ambiguity }
else if number of parses is greater than one and identical suffix ambiguity then
 call Resolve Identical Suffix Ambiguity
 if Resolved then
  output the translation
 else if not resolved then
  if the word also involves multiple root word ambiguity then
   call Resolve Multiple Root Words Ambiguity
   if resolved then
    output the translation
   else
    output the alternative translations
   end if
  else
   output the alternative translations
  end if
 end if
end if
end if
end for
end while
```

Figure 5.3. Main Algorithm (continued)

5.3.2. Modifying Translator Lexicons

In this routine, if the user supplies the correct password for a supervisor, he can modify the translator's lexicons. Namely, he modifies the lexicons in text file format, and transfers them into suitable file format with this routine. The algorithm is given in Figure 5.4.

request the password
if correct password then
read the name of the lexicons to be modified
for each lexicon read do
call the appropriate routine to transfer the lexicon from text file format into suitable file format that can be used by the translator
end for
end if

Figure 5.4. Algorithm for Modifying the Translator Lexicons

5.3.3. Turkish Morphological Parser

The Turkish morphological parser obtained from Boğaziçi University [8] is refined, and used as a separate unit which can be called by the translator.

This unit receives a Turkish word, and produces all of its morphological parses together with the related information. For each parse of the word the following information is obtained:

1. *the Turkish root word*
2. *its location in the Turkish root words lexicon*
3. *its part of speech,*
4. *its subcategories,*
5. *its semantic features,*
6. *the Turkish suffixes affixed to the word,*

7. their related information for the vowel harmony rule and the consonant harmony rules,

8. the generic names of the suffixes,

9. their Azeri equivalents,

10. the related information of the Azeri equivalents for the vowel harmony rule and the consonant harmony rules,

11. a flag to denote whether the root word is a proper noun or not,

12. a flag to denote whether the word involves identical suffix ambiguity or not.

This information is utilized to diagnose the kind of ambiguity the word involves and to resolve the ambiguity. For a detailed algorithm of this unit see "Internal Design Specification, A Spelling Checker and Corrector for Turkish" [24] .

5.3.4. Affixing Suffixes

The routine receives an Azeri root word and the Azeri suffixes that will be affixed to it. It forms the equivalent Azeri word by affixing the suffixes to the root. In order to affix a suffix to an Azeri word it uses Affix a Suffix routine defined inside it.

5.3.4.1. Affixing a Suffix

This routine affixes an Azeri suffix to an Azeri word applying the morphophonemic rules of Azeri. The algorithm is given in Figure 5.5.

```
modify the root word and the suffix by applying the following morphophonemic rules
 apply the vowel harmony rule
 apply the consonant harmony rules
affix the suffix to the root word
return the new form of the word
```

Figure 5.5. Algorithm for Affixing a Suffix

5.3.5. No Ambiguity Case

In case of no ambiguity, if the Turkish word is not a proper noun its equivalent Azeri word is found. If it is a proper noun, it is directly used as the Azeri equivalent. Then the translation is performed by affixing the Azeri equivalents of Turkish suffixes to the equivalent Azeri root word. The algorithm is given in Figure 5.6.

if the root word is not a proper noun then
find the Azeri equivalent of the Turkish root word from Translation lexicon
else
use the Turkish proper noun as the equivalent Azeri root word without any modification
end if
call Affix Suffixes in order to affix the Azeri equivalents of the Turkish suffixes identified for the Turkish word to the Azeri root word.

Figure 5.6. Algorithm for No Ambiguity Case

5.3.6. Finding Collocation Index

This routine receives the first element of a collocation definition from Check Collocation. It finds the places of the words in the sentence that satisfy this element. The algorithm is given in Figure 5.7. In the algorithm, each possible case for the collocation element is represented with its symbol (see Section 4.5.3). X denotes any possible value for the place it appears.

set all entries of the collocation index array to zero
if the collocation element is a macro collocation definition then
{ *coX* }
set the next collocation index array entry to minus one

Figure 5.7. Algorithm for Finding Collocation Index

else if it is a word with the part of speech then

{ *ca(X)* }

for all words of the sentence do

if the part of speech of the root word for any parse of it is the same as the part of speech X then

store the place of the word in the sentence to the next collocation index array entry

end for

end if

else if it is a word with the specific semantic feature then

{ *cX* }

for all words of the sentence do

if the root word for any parse of it has the semantic feature X then

store the place of the word in the sentence to the next collocation index array entry

end for

end if

else if it is a word with the proper noun semantic feature then

{ *oX* }

for all words of the sentence do

if the root word for any parse of it has the proper noun semantic feature X then

store the place of the word in the sentence to the next collocation index array entry

end for

end if

else if it is a word with the noun semantic feature then

{ *X* }

for all words of the sentence do

if the root word for any parse of it has the semantic feature X then

store the place of the word in the sentence to the next collocation index array entry

end for

end if

else if it is a word with the specified subcategory then

{ *a(X) or d(X) or p(X) or v(X)* }

for all words of the sentence do

if the root word for any parse of it belongs to the specified subcategory then

store the place of the word in the sentence to the next collocation index array entry

end for

Figure 5.7. Algorithm for Finding Collocation Index (continued)

```
end if
else if it is the ambiguous word then
{ - }
 for all words of the sentence do
  if the word is the same as the ambiguous word then
    if the location of the word in the sentence is equal to the collocation index then
     store the place of the word in the sentence to the next collocation index array entry
 end for
end if
else if it is a sentence with the mood then
{ mood(X) }
 if the mood of the sentence is the same with the specified mood X then
  set the next collocation index array entry to minus one
else if it is alternative collocation then
{ ( , ) }
 find each alternative collocation
 for each alternative collocation do
  for all words of the sentence do
   if the collocation is true then
    store the place of the word in the sentence to the next collocation index array entry
  end for
 end for
end if
else if it is a constant word then
{ e.g. güzel, geldi }
 for all words of the sentence do
  if the word is the same as the constant word then
   store the place of the word in the sentence to the next collocation index array entry
 end for
end if
return the collocation index array
```

Figure 5.7. Algorithm for Finding Collocation Index (continued)

Here a couple of traces might be useful to understand the algorithm better. Consider the Turkish sentence *Ali bir kalem ve silgi aldı (Ali bought a pencil and an eraser).* Assume that the first element of the collocation that is currently analyzed is *ca(n)* which means a word with part of speech noun. In this case, since the Turkish sentence has two words *(kalem, silgi)* satisfying this element, find collocation index routine returns their locations in the sentence, namely 3 and 5. These are the possible places from which the collocation can be started to check.

Another example might be the Turkish sentence *Beni seviyor (He/she loves me).* If the first element of the collocation is *ca(p)+accu* , which means a pronoun with an accusative suffix, only 1 is returned by the routine. This is because only the word *beni* in the sentence satisfies this element.

5.3.7. Checking Collocation

This routine checks whether the current sentence satisfies a collocation or not. If it satisfies returns true, otherwise returns false. The algorithm is given in Figure 5.8. In the algorithm, each possible case for the collocation element that is to be checked is represented with its symbol (see Section 4.5.3). X denotes any possible value for the place it appears.

find the distinct elements in the collocation
call Find Collocation Index to find the places of the words, that satisfy the first element of the collocation, in the sentence
set satisfied to false
for each word found and while not satisfied do
set okay to true
set collocation index as the location of the next word in the sentence
for all the distinct elements of the collocation beginning from the second one and while okay do

Figure 5.8. Algorithm for Checking Collocation

if it is a macro collocation definition then

{ *coX* }

call Check Macro Collocation Definition

if it returns false then

set okay to false

end if

else if it is a word with the part of speech then

{ *ca(X)* }

if none of the parses of the word located in the current collocation index has a root word with part of speech X then

set okay to false

end if

else if it is a word with the specific semantic feature then

{ *cX* }

if none of the parses of the word located in the current collocation index has a root word with semantic feature X then

set okay to false

end if

else if it is a word with the proper noun semantic feature then

{ *oX* }

if none of the parses of the word located in the current collocation index has a root word with semantic feature X then

set okay to false

end if

else if it is a word with the noun semantic feature then

{ *X* }

if none of the parses of the word located in the current collocation index has a root word with semantic feature X then

set okay to false

end if

else if it is a word with the specified subcategory then

{ *a(X) or d(X) or p(X) or v(X)* }

Figure 5.8. Algorithm for Checking Collocation (continued)

if none of the parses of the word located in the current collocation index has a root word which belongs to the specified subcategory then

set okay to false

end if

else if it is the ambiguous word then

{ - }

if the word located in the current collocation index is not same as the ambiguous word or if the location of the word in the sentence is not equal to the collocation index then

set okay to false

end if

else if it is the sentence with the mood then

{ *mood(X)* }

if the mood of the sentence is not the same with the specified mood X then

set okay to false

end if

else if it is alternative collocation then

{ (,) }

find each alternative collocation

if none of the alternative collocations is true for the word located in the collocation index then

set okay to false

end if

else if it is a constant word then

{ *e.g. güzel, geldi* }

if the word located in the current collocation index is not the same as the constant word then

set okay to false

end if

end for

if okay then

set satisfied to true

end for

return satisfied

end if

Figure 5.8. Algorithm for Checking Collocation (continued)

As an example for the trace of this routine, consider the Turkish sentence *yarın sabah gelecekler. (They will come tomorrow morning)* In this sentence the ambiguous word is *yarın* as it has five morphological parses:

yar(abyss) + ın(2nd singular possessive)
yar(abyss) + ın(genitive)
yar(to split) + ın(2nd singular person)
yarı(half) + n(2nd singular possessive)
yarın(tomorrow)

The following collocation of the word *yarın* will be checked by this routine to decide whether the Turkish sentence satisfies it or not:

- *(sabah, öğle, akşam, gece)*

Firstly, the elements of the collocation is determined. In this case there exist two elements: - and *(sabah, öğle, akşam, gece).*

Next, Find Collocation Index routine is called with the Turkish sentence and the collocation element -. It returns 1 since the ambiguous word *yarın* exist only as the first word of the sentence.

Finally, the remaining elements of the collocation are checked beginning from the second word of the sentence. The next element is *(sabah, öğle, akşam, gece)* which is an alternative collocation, and the second word of the sentence satisfies this collocation element. Since there exist just 2 elements in the collocation, the collocation definition is satisfied by the check collocation routine. Thus it returns true.

5.3.8. Checking Macro Collocation Definition

This routine gets a macro collocation definition number, and reads the collocation definition located in this entry of the Macro Collocation Info Definitions lexicon. It then checks whether any one of the collocations of this definition is satisfied by the Turkish sentence or not. Figure 5.9 gives the algorithm.

set satisfied to false
for each collocation designated in the specified macro collocation definition in the Macro Collocation Definitions Lexicon do
 if Check Macro Collocation Definition returns true then
  set satisfied to true
return satisfied

Figure 5.9. Algorithm for Checking Macro Collocation Definition

5.3.9. Resolving Word Sense Ambiguity

This routine tries to resolve the word sense ambiguity by utilizing the concept and collocation info structures. It first checks the collocation info structure, because of its high priority in comparison with the concept structure. It returns true if accomplishes to resolve the ambiguity, and returns false if not. The algorithm is given in Figure 5.10.

```
set resolved to false
for all word senses of the word and while not resolved do
 if collocation info structure is utilized for the word sense then
  for all the collocations of the word sense do
   call Check Collocation to check whether the Turkish sentence satisfies the collocation or not
   if it returns true then
    set resolved to true
  end for
 end if
end for
for all word senses of the word and while not resolved do
 if concept structure is utilized for the word sense then
  for all concepts the word sense is dependent on do
   if the concept is active then
    set resolved to true
  end for
 end if
end for
return resolved
```

Figure 5.10. Algorithm for Resolving Word Sense Ambiguity

5.3.10. Resolving Multiple Root Words Ambiguity

This routine tries to resolve the multiple root words ambiguity by utilizing the concept and collocation info structures. It first checks the collocation info structure, because of its high priority in comparison with the concept structure. If any of the ambiguous root words is also word sense ambiguous, then it calls Resolve Word Sense Ambiguity routine to resolve the ambiguity. It returns true if it accomplishes to resolve the ambiguity, and returns false if not. Figure 5.11 gives the algorithm.

```
set resolved to false
for all morphological parses of the word and while not resolved do
 find the corresponding entry in the Morphologically Ambiguous Root Words lexicon for the Turkish root word
of the parse
 if the root word is word sense ambiguous then
  call Resolve Word Sense Ambiguity
 else if collocation info structure is utilized for the word then
  for all the collocations of the word do
   if Check Collocation returns true
    set resolved to true
 end if
end for
for all morphological parses of the word and while not resolved do
 find the corresponding entry in the Morphologically Ambiguous Root Words lexicon for the Turkish root word
of the parse
  if concept structure is utilized for the word then
   if any of the concepts that the word is dependent on is active then
    set resolved to true
 end for
return resolved
```

Figure 5.11. Algorithm for Resolving Multiple Root Words Ambiguity

5.3.11. Resolving Identical Suffix Ambiguity

This routine tries to resolve the identical suffix ambiguity by utilizing the collocation info structure. It returns true if it accomplishes to resolve the ambiguity, and returns false if not. Even though the identical suffix ambiguity is resolved, the root word may be word sense ambiguous. In this case, if the identical suffix ambiguity is resolved, then the word sense ambiguity is treated accordingly. Figure 5.12 gives the algorithm.

```
set resolved to false
for all morphological parses of the word and while not resolved do
 find the corresponding entry in the Identical Suffixes lexicon for the ambiguous suffixes involved in the parse
 for all the collocations do
  call Check Collocation to check whether the Turkish sentence satisfies the collocation or not
  if it returns true then
   set resolved to true
 end for
end for
if resolved then
 if the root word is word sense ambiguous then
  call Resolve Word Sense Ambiguity
end if
return resolved
```

Figure 5.12. Algorithm for Resolving Identical Suffix Ambiguity

6. DISCUSSION AND EVALUATION

In this chapter the shortcomings and the performance of the translator implemented in this work are discussed, and proposals to improve its performance are put forward.

6.1. Shortcomings of the Translator

We are aware of the following shortcomings of the translator:

1. incomplete linguistic data
2. shortcomings of the Turkish morphological parser
3. translation of idioms and expressions
4. shortcoming about the concept structure
5. ambiguities caused by proper nouns
6. restriction on the number of words of a sentence

The most obvious shortcoming of the translator is its incomplete linguistic data. It covers 6900 Turkish words and their Azeri equivalents. The information used to resolve the ambiguities does not include all the co-occurrence data for ambiguous Turkish words, since they are obtained from dictionaries and from the linguistic knowledge of the author instead of real Turkish corpora. In order to collect the complete co-occurrence data for ambiguous Turkish words, a detailed corpus analysis for Turkish is necessary.

The Azeri equivalents of a small number of the Turkish derivational suffixes can not be determined from the linguistic references. These suffixes are the following:

-(y)Adur, -(y)Agel, -(y)Agör, -(y)Akoy, -(y)Iver, -(y)Ayaz, -cAsInA,
-trilyon, -cAğIz, -(I)z (for numbers), -sIzIn, -(y)IlAn, -(y)InAn,
-(y)AlI, -(y)AsIyA

The translator treats these suffixes as if they are the same with their Azeri equivalents.

Since the translator diagnoses the ambiguities using the results obtained from the Turkish morphological parser developed at Boğaziçi University, all of its shortcomings cause problems in the translator, e.g. wrong parses. If a word can be morphologically parsed in more than one way, it is identified as ambiguous. Hence, in case of wrong parses, the Turkish word is incorrectly identified as ambiguous. Since there exists no information to resolve the ambiguity, it can not be resolved and the result of the translation is given as alternative translations, which is actually not the case.

The translator does not handle expressions and idioms appropriately. Although some expressions are placed in the root lexicon, it simply ignores them. This may cause wrong translation as in the following example. The Azeri equivalent of the Turkish expression *hoşa gitmek* is *xoşa gälmäk*. The translator first translates the word *hoşa* as *xoşa* and then the word *gitmek* is translated as *getmäk*, since the Azeri equivalent of the Turkish word *gitmek* is *getmäk*. So in this way the expression *hoşa gitmek* is wrongly translated as *xoşa getmäk*.

Another shortcoming of the translator is about the concept structure. In the current implementation only root words are allowed to activate the concepts. This causes the following problem. If a derived word activates a concept even though its root does not activate the same concept, this information can not be coded into the translation lexicon. These words should be treated separately. They should be determined and stored in a separate lexicon, and should be allowed to activate the related concepts.

Another shortcoming of the translator is related to proper nouns. It does not handle ambiguities caused by proper nouns. It simply assumes that words beginning with a capital letter are proper nouns, and therefore it requires all other words to begin with the lower case letters, even for the first word of a sentence. So, in the sentence *dün Aydın geldi.*, *Aydın* is treated as a proper noun, as in the sentence *Aydın bir toplum*

olmalıyız. This causes incorrect translation for the second sentence. Therefore to avoid this kind of wrong translations, it should be ensured that the input text does not contain any capital letters except for proper nouns. This restriction can be released by checking the first word of the sentence to identify whether it is an ambiguous word of this kind or not. In case of an ambiguity, it can be resolved using the concept and collocation info structures.

Finally, we should mention the following limitation of the translator. The current implementation expects at most 10 words in a sentence. This is due to the limit imposed on accessible main memory by the Turbo Pascal 6.0 compiler, in comparison with the enormous amount of code and data size of the translator.

6.2. Performance Evaluation

As we discussed in Section 2.1, there exist some performance measures proposed for evaluating the quality of machine translation systems. Since, we did not test our translator using real Turkish corpora, we could not perform a thorough performance evaluation. However, the performance evaluation of the translator considering the results obtained so far (see Appendix F) according to the performance metrics presented in Section 2.1 is as follows:

1. *Linguistic generality*: The translator involves just one source and the target language. The extent of coverage in the vocabulary is 6900 words, and the idioms are not taken into consideration.

2. *Application domain generality*: The translator does not involve the use of any sublanguages, and the subject domain covers all the language.

3. *Degree of automation*: Although, it does not involve any human intervention during the translation process, the results produced may in some cases need postediting. However, we did not measure the amount of time required for human intervention.

4. *Semantic accuracy*: The results produced by the translator were not examined by an Azeri native speaker, so its semantic accuracy is not tested.

5.*Intelligibility*: Since, the results of the translator were not examined by an Azeri native speaker, its intelligibility is not tested.

6. *Appropriateness*: The translator is not tested according to this criterion.

7. *Domain and language portability*: Because of its modular structure, other Turkic languages can be integrated into the translator without much difficulty.

8. *Extensibility*: The lexical coverage of the translator for unambiguous words can be extended easily, since the structure of the lexicons and the interrelation among the information in the lexicons are clear. However, addition of ambiguous words may need extra treatment, e.g. defining new semantic features, defining new concepts, and determining collocation information of the ambiguous words.

9. *Improvability*: As the lexical knowledge improves, the quality of the translator also improves without any redesign effort.

10. *Ergonomics*: Since the translator already has a relatively simple user interface, it seems quite ergonomic.

11. *Integrability*: The translator can be integrated into other information processing applications without much difficulty.

12. *Software portability*: The translator can be ported to other hardware platforms as long as its minimum hardware requirements are met.

In addition to these criteria, we would like to say a couple of things about the speed of the translator, since a machine translator should also be able to translate in a reasonable amount of time. The basic factor influencing the speed of the translator is the Turkish morphological parser embedded in it. The words in the input text are first morphologically parsed using this parser. So, if the time it takes for the morphological parser to find the morphological parses of a word is tmp1, and the rest of the process takes a time of tmp2, then the translation time for this word is tmp1 + tmp2.

The second factor is the number of ambiguities in the input text. If a word is not ambiguous, then the time it takes for its translation is slightly more than tmp1. However, in case of ambiguity translation time increases considerably, especially if the resolution process involves the use of collocation info structure. The critical routine employed in the resolution of ambiguities is the check collocation routine which implements the collocation info structure. The time needed for resolving ambiguity is highly dependent on this routine.

The final factor affecting the speed of the translator is its memory requirement. The main program is 270 K bytes together with the Turkish morphological parser. In the current implementation, in the case of loading all the data files into main memory, the memory requirements for the static and the dynamic variables are approximately as given in Table 6.1.

Table 6.1. Memory requirement of the translator

The structure	Size(in K bytes)
Turkish root words lexicon	690
Turkish proper nouns lexicon	600
translation lexicon	280
word sense ambiguous root words lexicon	125
morphologically ambiguous root words lexicon	120
bilingual suffix lexicon	10
identical suffixes lexicon	9
parse results for the sentence under translation	6
macro collocation info definitions lexicon	4
root words lexicon and proper nouns lexicon indexes	4
miscellaneous	1

So, in case of full memory utilization the translator needs approximately 2MB memory for its data files. Since, we have implemented the translator using Turbo Pascal 6.0 compiler, we can not achieve full memory utilization. So, we are restricted to use file access. This restriction causes a considerable amount of time loss during data access which lowers the speed of the translator.

In order to determine the real speed of the translator, it should be checked with a large enough sample text. However, we tested the translator with simple sentences that contain just the ambiguities we introduce to check the correctness of the translator. Sample runs for the translator listed in Appendix F were obtained using an IBM compatible computer with a 80386 processor running at 25 Mhz. The following is an example of these sample runs:

Turkish text: [25]

yatağın başından ucuna kadar uzanan mavi damalı yorganın
engebeleri gölgeli vadileri ve mavi yumuşak tepeleriyle örtülü
tatlı ve ılık karanlıkta rüya yüzü koyun uzanmış uyuyordu.
dışarıdan kış sabahının ilk sesleri geliyordu.

Azeri text:

uyadakın başından ucuna qädär uzanan mavi damalı yorğanın eniş-yoxuşları kölgäli vadiläri vä mavi yumşag täpäläriylä hörtülü dadlı vä ılıg garanlıkda rö'ya (yüz / üz) (goyun / gatı - ın / buxta - ın / buxta - nın) uzanmış uyurdu. dışarıdan gış sähärinin ilk säslări gälirdi .

We have separated the input text into 10 words sentences before the translation process. Since the translation is carried out sentence by sentence, all the collocation info in the first sentence can not be utilized in the translation. This causes semantic information loss

for the ambiguity resolution process. The words in the Turkish expression *yüzü koyun* are translated one by one, since they are written separately in the input text. Translator identified the Turkish word *yüzü* as ambiguous, since the root word *yüz* has two word senses, *hundred* and *face*. Since it could not resolve the ambiguity, the two alternatives are given in the translation. Although the word *rüya* is used as proper noun in the input text, since its first letter is not a capital letter, it is not treated as a proper noun and translated incorrectly. The word *koyun* is also identified as ambiguous, since it has the following five morphological parses:

koyun(sheep)
koyu(dense) - n(2nd singular possessive)
koy(inlet) - un(2nd singular possessive)
koy(inlet) - un(genitive)
koy(to put) - un(2nd plural person)

Since the translator could not achieve to resolve the ambiguity, all the alternatives are given in the translation. Since, the Azeri equivalents of the first and the fifth parses are the same, four distinct alternatives are presented in the translation.

6.3 Further Improvements for the Translator

As we stated previously, in some cases syntactic analysis may help in resolving ambiguities. For example, if the subject and the verb of the sentence are determined, the subject-verb agreement rule may be utilized in ambiguity resolution. Therefore, a syntactic analyzer for Turkish may be developed and embedded into the translator to improve its performance on resolving ambiguities.

In the current implementation, the translator checks all the previous sentences together with the current sentence to resolve the ambiguity. Another improvement for the

translator may be to check succeeding sentences in the input text to resolve ambiguities. For instance in the Turkish text *altı güzel değil. yediyi seçtim.*, the Turkish word *altı* is ambiguous as *alt(bottom)* - *ı(accusative/3rd singular possessive)* and *altı(six)*. The ambiguity can not be resolved with the semantic information obtained from the first sentence. However, if the second sentence is also utilized in ambiguity resolution, *altı(six)* would be identified as the correct meaning.

An additional routine to automate the addition of a Turkish word into translator's lexicons may help to reduce the time needed for this process, and decreases the possibility of introducing errors. In case of such an addition, first the word should be checked for word sense ambiguity and then it should be checked against all the root words lexicon in order to decide whether it causes any multiple root words ambiguity or not. If it involves an ambiguity, the related information for the disambiguation process should be provided. This may involve defining new semantic features, and determining the collocation information of the word. Then, the new word should be checked to decide whether it activates any concept or not, and the necessary information should also be provided accordingly. If a new concept should be defined and associated with the word, then all the words in the root words lexicon that would activate this concept should be determined, and this information should be added to the lexicon. Finally, the semantic features, parts of speech and the morphological characteristics of the word should be added to the root words lexicon.

An intelligent routine may be designed and embedded into the translator to learn the co-occurrence information when ambiguities could not be resolved by the translator. In this manner, the linguistic content of its lexicons can be automatically augmented.

7. FURTHER DEVELOPMENTS AND CONCLUSION

Using the translator implemented in this work and the proposed approach, a machine translation system from Azeri to Turkish, and a machine translation system from Turkish to all the other Turkic languages can be realized as further developments.

7.1. Conclusion

Since the existing approaches are not suitable for the problem of machine translation from Turkish to other Turkic languages, we propose a lexicon-based approach. As the sentence syntax is similar for both languages we do not employ a syntactic analysis. Morphological and semantic analysis are carried out for the translation. The translation is performed as a word for word translation of sentences using the disambiguation techniques used in transfer-based and interlingua-based translation approaches.

As this is the first attempt in this field the problems of the subject and possible ways to handle them are put forward. The ambiguity subject in translation from Turkish to other Turkic languages is explored, different sorts of ambiguities are investigated, and the lexical data that cause those ambiguities are identified. Possible ways for ambiguity resolution are investigated. So, the translation is achieved by direct translation of unambiguous words and with a special treatment of the ambiguous ones. The contents of the lexicons in the translation system are determined, and possible lexicon structures are investigated. Finally, a practical translation system is developed with the proposed approach to evaluate its feasibility.

Even though the results seemed satisfactory for the feasibility of the approach, improvements might be implemented to improve the performance and the quality of the program. Moreover for its testing, a detailed performance analysis is needed by applying real input texts.

7.2. Lexicon Formation by Corpus Analysis

The most essential part of a translator is its lexicons. A translation system should have well designed lexicons with complete linguistic data. The most suitable way to collect co-occurrence information of words is corpus analysis. So, an important task in constructing a translation system for Turkish is forming a lexicon by analyzing real Turkish corpora. A corpus analyzer may be designed and implemented to automate this process. In this way, co-occurrence information for ambiguous words may be obtained from the Turkish texts in a structured manner.

7.3. Towards a Machine Translation System from Azeri to Turkish

We have actually worked on the problem of machine translation from Turkish to Azeri, and not dealt with translation from Azeri to Turkish. Consequently, the translator developed does translation in one direction, only from Turkish to Azeri.

However, a translator from Azeri to Turkish can easily be implemented using the same method. First, an Azeri root words lexicon should be constructed together with its corresponding translation lexicon for Turkish words. Then the ambiguous Azeri words should be determined and treated in the same manner. The Turkish morphological parser can be used to parse Azeri words with small modifications. Finally, the same translator program can be used for translation from Azeri to Turkish, with the lexicons prepared for the Azeri language.

A machine translation system from Azeri to Turkish can be utilized to test the semantic accuracy of the translator developed in this work.

7.4. Towards a Machine Translation System from Turkish to Other Turkic Languages

Since in the translation system the data and the code is separated as much as possible, and it is designed in a modular structure, another Turkic language can be integrated into it without much difficulty.

First, the bilingual suffix lexicon should be modified to include the suffixes of the new language, and the morphophonemic rules of the language should be determined. A translator lexicon, in the same way as for the Azeri language, should be constructed for its root words.

Then the ambiguous Turkish words stored in the lexicons of the translator and the ones that have been ignored in translation from Turkish to Azeri should be checked to eliminate the ambiguities that can be ignored in the disambiguation process. Using the remaining ambiguous words the related lexicons for the ambiguity resolution process should be constructed.

Finally, the translator program should be modified to process the new lexicons developed for the new language, and a routine for affixing suffixes to stems using the morphophonemic rules of the language should be added to it.

In this way, with the integration of all the Turkic languages into the current translation system, a machine translation system from Turkish to other Turkic languages can be realized.

APPENDIX A. LIST OF WORD SENSE AMBIGUOUS WORDS

alay	aşı	ayar	bağlı
baskı	başıboş	ben	bez
cereyan	cetvel	cevher	cilt
cümle	çağdaş	çapa	çat
çatal	çekim	çerçöp	çevir
çırp	çiğne	daire	dal
dal	darbe	dava	dayak
dayı	değerlendir	deneme	denk
derece	derman	dik	divan
diye	diyet	dizi	doğ
don	dosya	duvar	düş
düzen	efendim	eğe	eğitim
eğlen	eğreti	ek	el
emsal	en	er	esne
eş	etek	fail	fen
fener	fıkırda	fıkra	fiş
fişek	garip	gebe	göbek
gövde	gözle	güç	hak
han	harç	hava	havale
hayır	hesaplı	hoca	horla
hortum	hücre	içik	ramiye
iktidar	ilahi	ilik	illet
işlet	mekalın	kalkan	kalp
kanun	kap	kara	kavra
kıy	koca	kol	kompleks
kon	kredi	kurşun	kurum
kuşak	kuyruk	lisans	makam
mâni	metin	muhtar	nasılsa
neden	nefis	not	ocak
olgun	olumlu	olumsuz	oyna
oyun	pek	pirinç	pişkin
pul	saf	sap	satır
saz	sessiz	sıfat	sıra
sıva	sinir	şekerleme	şık
taban	tabir	takım	tekne
temsil	tezgâh	tokmak	tulum
tuş	ünlü	vasat	vekâlet
vur	yaka	yalı	yapı
yargı	yazı	yüz	yüzmek
zar			

APPENDIX B. LIST OF MORPHOLOGICALLY AMBIGUOUS WORDS

B.1. List of Root Words that are Similar in Form to Stems with Inflectional Suffixes

The ambiguous root words for each suffix is presented below. The symbol (v) is used to denote the verbs.

Suffix : -(y)I -(s)I

The Ambiguous Words :

bileği	duyu
bilek	duy
koru	koyu
kor	koy
resmi	solu(v)
resim	sol
açı	askeri
aç	asker
aşı	dişi
aş	diş
dizi	eri(v)
diz	er
karı	katı
kar	kat
martı	salı
mart	sal

Suffix : -(y)A

The Ambiguous Words :

ada	ata
ad	at
bana	benze(v)
ban(v)	beniz
boğa	çıta
boğ(v)	çıt
dene(v)	deve
de(v)	dev
devre	doğa
devir	doğ
eğe	ele(v)
eğ(v)	el
hale	halka
hal	halk
ilke	kala
ilk	kal(v)
kaya	kına
kay(v)	kın
kota	koza
kot	koz
kuma	küfe
kum	küf
ova	öte
ov(v)	öt(v)
rehine	semere
rehin	semer

süre	tarife
sür(v)	tarif
türe	ufala(v)
tür	ufal(v)
vezne	yaka
vezin	yak(v)
yıka(v)	
yık(v)	

Suffix : -DA

The Ambiguous Words :

hasta	hatta
has	hat
iste(v)	işte
is	iş
sapta(v)	gözde
sap	göz

Suffix : -(I)n

The Ambiguous Words :

basın	demin
bas(v)	dem
eşin(v)	halen
eş	hale
havan	kalın
hava	kal(v)

kalkın(v)	kayın
kalk(v)	kay(v)
---	---
memnun	ödün
memnu	öd
---	---
peşin	sahan
peş	saha
---	---
somun	sorun
som	sor(v)
---	---
sultan	tosun
sulta	tos
---	---
tören	tütün
töre	tüt(v)
---	---
yazın	
yaz(v)	

Suffix : -(I)m

The Ambiguous Words :

---	---
devrim	dilim
devir	dil
---	---
doğum	dönüm
doğu	dön(v)
---	---
hanım	ilim
han	il
---	---
kalem	kasım
kale	kas
---	---
kesim	koşum
kes(v)	koş(v)
---	---

meram	rakım
mera	rakı
---	---
terim	verim
ter	veri
---	---

Suffix : -(y)Iz

The Ambiguous Words :

---	---
filiz	hırsız
fil	hırs
---	---
otuz	
ot	

Suffix : -DI

The Ambiguous Words :

---	---
girdi	türedi
gir(v)	türe(v)
---	---
uydu	yedi
uy(v)	ye(v)
---	---

Suffix : -mIş

The Ambiguous Words :

---	---
dolmuş	yemiş
dol(v)	ye(v)
---	---
yetmiş	
yet(v)	

Suffix : -Ar

The Ambiguous Words :

der(v)	yer
de(v)	ye(v)
eğer	kaçar
eğ(v)	kaç(v)
uyar	yakar
uy(v)	yak(v)

Suffix : -mA

The Ambiguous Words :

çizme	inme
çiz(v)	in(v)
kıyma	saçma
kıy(v)	saç(v)
yağma	
yağ(v)	

Suffix : -lA

The Ambiguous Words :

ağla(v)	atla(v)
ağ	at
çatla(v)	emekle(v)
çat	emek

gülle	inle(v)
gül	in
kelle	sağla(v)
kel	sağ
salla(v)	sapla(v)
sal	sap
taklaya	sakla(v)
takya	sak

Suffix : -sA

The Ambiguous Words :

ense	hassa
en	has
kıssa	
kıs(v)	

Suffix : -mAk

The Ambiguous Words :

çakmak	kaymak
çak	kay
oymak	
oy	

Ambiguous words for miscellaneous suffixes :

aralık	aşama
ara	aş(v)
bildik	çekici
bil(v)	çekiç(v)
değişim	desen
değ(v)	de(v)
dolandır	elli
dol(v)	el
etken	geçende
et	geçe
girişim	içişleri
gir(v)	iç(v)
kalıp	kapalı
kal(v)	kap(v)
kaplan	kaptan
kap	kap
kayıp	kelime
kay(v)	kel
koşul	oluşum
koş(v)	ol(v)
seciye	seksen
seci	sek(v)
tandır	tarihçe
tan	tarih
telsiz	yaşasın
tel	yaşa(v)
yoksun	zarfında
yok	zarf

B.2. List of Root Words that are Similar in Form to Stems with Derivational Suffixes

Suffix : -(I)k

The Ambiguous Words :

buruk	kasık
bur(v)	kas(v)
kayık	yayık
kay(v)	yay(v)

Suffix : -(I)ş

The Ambiguous Words :

buluş(v)	buruş(v)
bul(v)	bur(v)
değiş(v)	eriş(v)
değ(v)	er(v)
kayış	konuş(v)
kay(v)	kon(v)
oluş(v)	tartış(v)
ol(v)	tart(v)
yakış(v)	yapış(v)
yak(v)	yap(v)
yatış(v)	yetiş(v)
yat(v)	yet(v)
yılış(v)	
yıl(v)	

Suffix : -CA

The Ambiguous Words :

karaca	sivilce
kara	sivil

Suffix : -(y)An

The Ambiguous Words:

alan	bakan
al(v)	bak(v)
diken	esen
dik(v)	es(v)
kalkan	kapan
kalk(v)	kap(v)
sapan	uyan(v)
sap(v)	uy(v)
yılan	
yıl(v)	

B.3. List of Ambiguous Words due to Similar Suffixes

The ambiguous words with the suffixes causing the ambiguity are presented below. Even though, in some cases there may actually more than one suffix causing ambiguity for the same words, only one of them is presented below and the rest is eliminated.

ajans	-a	akıl	-ı
ajan	-sa	ak	-lı
asil	-i	dem	-e
asi	-li	de	-me
deli	-ın	deney	-e
del	-ın	dene	-ye
derle	-r	eksi	-ın
de	-r -ler	ek(v)	-sın
		ek	-sın
kalas	-ın	kapı	-ın
kal	-a -sın	kap(v)	-ın
kurmay	-a	lisans	-a
kur	-ma -ya	lisan	-sa
oy	-sa	pirinç	-e
o	-ysa	pir	-in -ce
saray	-a	sağlam	-a
sara	-ya	sağla(v)	-ma
örs	-e		
ör(v)	-se		

APPENDIX C. LIST OF MORPHOLOGICAL PARSES PRODUCED BY IDENTICAL SUFFIXES

The morphological parses for identical suffixes obtained from the Turkish morphological parser are presented below. In each parse, first the artificial root word, the word after affixing the suffix, and the root word category are given. Then the parses obtained for the word are given as transitions between word categories in the transition network representation. - corresponds to a free jump, and a suffix name between two word categories denotes the transition between them with this suffix. Each transition is represented with the final word category and the suffix causing the transition, e.g. n1 - means the transition from previous word category to n1 with free jump.

---

a alan n0
n1 - v9 lan
a0 - v8 al v10 - v16 - v18 - d2 yan
a0 - v8 al v10 - v16 - v18 - v73 - v19 - v24 - v25 - a5 yan

---

ad adı n0
n1 - n6 - n7 - n9 - n18 - n11 - n20 yı
n1 - n6 - n7 - n9 - n22 - n17 sı

---

at atım n0
n1 - n6 - n7 - n9 - n10 ım
n1 - n6 - n7 - n9 - n18 - n11 - n16 - v44 - v45 - v48 yım

ı ınız n0
n1 - n6 - n7 - n9 - n10 ınız
n1 - n6 - n7 - n9 - n10 ın n18 - n11 - n16 - v44 - v45 - v48 yız
n1 - n6 - n7 - n9 - n18 - n11 - n15 nın n16 - v44 - v45 - v48 yız

---

at atın n0
n1 - n6 - n7 - n9 - n10 ın
n1 - n6 - n7 - n9 - n18 - n11 - n15 nın

---

a anın n0
n1 - n6 - n7 - n9 - n10 ın n18 - n11 - n15 nın
n1 - n6 - n7 - n9 - n18 - n11 - n15 nın

---

ed eden n0
a0 - d2 dan
n1 - n6 - n7 - n9 - n18 - n11 - n16 dan

---

ad adır v2
v3 ır
v16 - v18 - v73 - v19 - v24 - v25 - n0 ır
v16 - v18 - v73 - v19 - v24 - v25 - v42 ır

---

ad adın v7
v12 ın
v16 - v18 - v73 - v19 - v24 - v25 - v79 yın

---

ad adış v7
v13 ış
v16 - v18 - v73 - v72 - v53 yış

---

a agın v1
v16 - a0 gın
v16 - n0 gı n1 - n6 - n7 - n9 - n10 ın
---
at atıcı v3
v10 - v16 - n0 ı a3 cı
v10 - v16 - n0 ı n5 cı
v10 - v16 - a3 yıcı
---
at atım v1
v16 - n0 ı n1 - n6 - n7 - n9 - n10 ım
v16 - v18 - n0 ım
---
a ama v1
v16 - v18 - n0 ım n1 - n6 - n7 - n9 - n18 - n11 - n14 ya
v16 - v18 - v72 ma
v16 - v18 - v73 - v19 - v21 - v54 ma
v16 - v18 - v73 - v19 - v24 - v25 - v30 ma
---
e eyen v1
v16 - v18 - d2 yan
v16 - v18 - v73 - v19 - v24 - v25 - a5 yan
---
e eye v1
v16 - v18 - v73 - v19 - v21 ya v78 -
v16 - v18 - v73 - v19 - v24 - d2 ya
v16 - v18 - v73 - v19 - v24 - v25 - v69 ya
v16 - v18 - v73 - v19 - v24 - v25 - v77 ya
---
e eyesiye v1
v16 - v18 - v73 - v19 - v24 - d2 yasıya
v16 - v18 - v73 - v19 - v24 - v25 - a5 yası n6 - n7 - n9 - n18 - n11 - n14 ya
---
a ar v1
v16 - v18 - v73 - v19 - v24 - v25 - n0 ır
v16 - v18 - v73 - v19 - v24 - v25 - v42 ır
---
a adık v1
v16 - v18 - v73 - v19 - v24 - v25 - v29 dık
v16 - v18 - v73 - v19 - v24 - v25 - v32 dı v34 k
---

a amakta v1
v16 - v18 - v73 - v19 - v24 - v25 - v43 makta
v16 - v18 - v73 - v19 - v24 - v25 - v31 mak n0 - n1 - n6 - n7 - n9 - n18 - n11 - n12 da
---
a amalı v1
v16 - v18 - v73 - v19 - v24 - v25 - v43 malı
v16 - v18 - v73 - v19 - v24 - v25 - v30 ma n0 - n1 - n6 - a0 lı
---
a amış v1
v16 - v18 - v73 - v19 - v24 - v25 - a0 mış
v16 - v18 - v73 - v19 - v24 - v25 - v36 mış
v16 - v18 - v73 - v19 - v24 - v25 - v43 mış
---
e eyecek v1
v16 - v18 - v73 - v19 - v24 - v25 - a5 yacak
v16 - v18 - v73 - v19 - v24 - v25 - v43 yacak
---
at atınca v3
v10 - v16 - n0 ı n1 - n6 - n7 - n9 - n10 ın a5 ca
v10 - v16 - n0 ı n1 - n6 - n7 - n9 - n10 ın d2 ca
v10 - v16 - v18 - v73 - v19 - v24 - v25 - d2 yınca
---
at atınız v2
v16 - n0 ı n1 - n6 - n7 - n9 - n10 ınız
v16 - n0 ı n1 - n6 - n7 - n9 - n10 ın n18 - n11 - n16 - v44 - v44 - v45 - v48 yız
v17 - v18 - v73 - v19 - v24 - v25 - v79 yınız
---
at atın v3
v10 - v16 - n0 ı n1 - n6 - n7 - n9 - n10 ın
v10 - v16 - v18 - v73 - v19 - v24 - v25 - v79 yın
---
ad adız s3
s3 - s4 - s24 - a3 ız
s3 - s4 - s24 - a3 - n0 - n1 - n6 - n7 - n9 - n18 - n11 - n16 - v44 - v45 - v48 yız
---

APPENDIX D. LIST OF THE SPECIFIC SEMANTIC FEATURES

Specific semantic features defined and used in the current implementation of the translator are presented below together with the numeric codes assigned to them. These codes are used in the translator's lexicons in order to refer to them. Since, in the lexicons, the numeric codes for these semantic features and the concepts are stored together in the same field, their numeric codes are assigned as a whole. Therefore, the numeric codes for them are not sequentially ordered.

Semantic feature	Numeric code
adjustable device	4
press	6
instrument to tie two concrete objects	7
trend	11
object that can be bound	12
drink	22
food	23
place like street, alley or region	28
press that can be criticized	29
sewing or material that can be sewed	31
name of a place that can be tidied up	32
object that can rise like sun and moon	33
object with a wall like a garden	35
time unit	36
food that can be cooked	46
fish or fish names	49
mode	50
object with an handle	51
word that the word mature can modify in a noun phrase	52
word that the words positive or negative can modify in a noun phrase	53
game played with a team	54
word that the word complex can modify in a noun phrase	55
object that can be weaved with a loom like carpet	57
place in which a queue of people may exist	59
device with keys like piano	60
gun	61
place that can be trampled down	63
animate or inanimate that can fly	65
inanimate that can not fly	66
objects that can be hard	67
ministry	69
weapon	70
foot and footwear	72
object that has a foot like mountain	73
water or container to warm water	74
human characteristic	75
rank name	76
fist type	77
clothes with a collar	78
food that can be beaten like egg	79
place that can be official	81
place that can be military	82

material that can melt 84
thing that can be performed in a grove 85
word that the word civil can modify in a noun phrase 86
object that can fill up 89
money unit 90
word that the word efficient can modify in a noun phrase 93
synonym of saying 94
type of science 95
places that can be measured with *dönüm* (a land measure of 1000 square meters) 96
thing that can blow like wind 98
thing that can be said 99
object that can be sliced 102
object that can be driven 106
object that can be washed 107
object that can be demolished 108
object that may have an opening like door 109
thing that has an answer like question or problem 110
thing that has a measurement unit like length 111
religion name 112
thing that can bleed 113
object that can be winded like watch or that can be set up like table 114
place characteristic 115
thing that can be turned into like road 116
material that can be used in sling 117
synonym of going or coming 118
object that can be ground like a knife 119
object that can be broken down 121
synonym of throwing 122
goods that can be sold in the market 123
writing type or writing characteristic 124
cliff characteristic 125
idea characteristic 126
object that can be worn away 127

APPENDIX E. LISTING OF THE LEXICONS

E.1. Representative Listing of the Turkish Root Words Lexicon

A representative listing of the Turkish root words lexicon, including its first 150 entries, is presented below in text form. Fields in an entry are separated by commas, and the information for each entry is given in a single line. The fields in each entry are in the following order:

1. Turkish word
2. parts of speech
3. flags
4. noun and pronoun semantic features
5. subcategories

0 and 1 represent the false and true boolean values, respectively. Noun semantic features are in the following order:

animate, inanimate, human, animal, plant,
concrete(0) / abstract(1), countable(0) / uncountable(1)

The third entry also serves for the pronoun semantic feature human interchangeably. The symbols used for subcategories are as follows:

reflexive pronoun(p1), personal pronoun(p2), demonstrative pronoun(p3), indefinite pronoun(p4), interrogative pronoun(p5), pronominal pronoun(p6), qualificative adjective(a1), demonstrative adjective(a2), interrogative adjective(a3), numeral adjective(a4), indefinite adjective(a5), time adjective(a6), time adverb(d1), place

adverb(d2), quantity adverb(d3), quality adverb(d4), interrogative adverb(d5), demonstrative adverb(d6), transitive(v1), intransitive(v2), dative(v4)

Qualificative adjective (a1) is the default value for adjectives. If no other subcategory information is specified for an adjective, it is understood that its subcategory is a1.

abajur	,n	,0000001000010010000000000000000001000	,0100000000	,
abart	,v4	,1000100000100000100	,0000000000	,v1 v4
abdest	,n	,0000000000010000000000000000000000000	,0000010000	,
abide	,n	,0000001000010010000000000000000001000	,0100000000	,
abla	,n	,0000000000010010000000000000000000000	,1010000000	,
abluka	,n	,0000000000000000000000000000000000000	,0000011000	,
abone	,n	,0000000000010000000000000000000000000	,1010000000	,
acayip	,i	,1000000000000000000	,0000000000	,
acayip	,a	,0000000000010000000000000000000000000	,0000000000	,
acele	,n a	,0000001000000000000000000000000001000	,1010000000	,
acele	,d	,1000000000000000000	,0000000000	,d1
acemi	,a	,0000000000010000000000000000000001000	,0000000000	,
acı	,v5	,1010000000000000100	,0000000000	,v2
acı	,n a	,0000000000010000000000000000000001000	,0000011000	,
acık	,v4	,1100000000000000100	,0000000000	,v2
acıklı	,a	,0000000000000000000000000000000000000	,0000000000	,
acımasız	,a	,0000000000010000001001000000000000000	,0000000000	,
acıt	,v4	,1000000000000000100	,0000000000	,v2
ad	,n	,0000000000000110010000000000000000000	,0000010000	,
ada	,v5	,1000000000000000000	,0000000000	,v1 v4
ada	,n	,0000000000000000000000000000000000000	,0100000000	,
adale	,n	,0000000000000010010000000000000001000	,0100000000	,
adalet	,n	,0000000000000000000000010000000000000	,0000010000	,
adam	,n	,0000000001010010000000000000000001000	,1010000000	,
aday	,n	,0000000000000000000000000000000000000	,1010000000	,
adaş	,n	,0000000000010000000000000000000000000	,1010000000	,
adi	,a	,0000000000010000000000000000000000000	,0000000000	,
adil	,a	,0000000000000000000000000000000000000	,0000000000	,
adres	,n	,0000000000000010110000000000000000000	,0000010000	,
adım	,n	,0000000000010010100000000000000000000	,0000010000	,
af	,n	,0000000000010110000000000010000000000	,0000010000	,
aferin	,n	,0000000000000010000000000000000000000	,0000010000	,

aferin	,i	,1000000000000000000	,0000000000	,
afet	,n	,00000000000100100000000010000000000000000	,0000010000	,
affet	,v4	,1000000000000100000	,0000000000	,v2
afiyet	,n	,00000000000000000000000010000000000000000	,0000010000	,
afiş	,n	,00000010000100100000000000000000000001000	,0100000000	,
afyon	,n	,00000010000000100000000000000000000001000	,0100001000	,
ahenk	,n	,00000000000100000000000010000000000000000	,0000010000	,
ahize	,n	,00000010000100100000000000000000000000000	,0100000000	,
ahlak	,n	,00000011000000000000000100000000000000000	,0000010000	,
ahmak	,a	,00000000000100000000000000000000000000000	,0000000000	,
ahret	,n	,00000000000100000000000100000000000000000	,0000010000	,
ahır	,n	,00000010000000100000000000000000000001000	,0100000000	,
aile	,n	,00000001000100100000000000000000000001000	,1010000000	,
ait	,e	,1000000000000000000	,0000000000	,
ajan	,n	,00000000000100100000000000000000000001000	,1010000000	,
ajans	,n	,00000000000100100000000000000000000000000	,0100000000	,
ak	,v6	,1000001000110000000	,0000000000	,v1
ak	,n a	,00000000000100000000000010000000000000000	,0000010000	,
akciğer	,n	,00000010000000100000000000000000000001000	,0100000000	,
akla	,v5	,1010000000000000000	,0000000000	,v2
akraba	,n	,00000000000100100000000000000000000001000	,1010000000	,
akrep	,n	,00000000000000000000000000000000000000000	,1001000000	,
aksa	,v5	,1000000001000000000	,0000000000	,v1
aksakal	,n	,00000000000000000000000000000000000000000	,0100001000	,
akset	,v4	,1000000000000100000	,0000000000	,v1 v4
aksi	,a	,00000000000100000110000000000000000000000	,0000000000	,
aksır	,v8	,1000000001000000100	,0000000000	,v1
aktar	,v3	,1000000000100000100	,0000000000	,v2 v4
aktarma yap	,v4	,1000000000000000000	,0000000000	,v1 v4
aktif	,a n	,00000000000000000000000000000000000000000	,1010000000	,
aktris	,n	,00000000000000000000000000000000000000000	,1010000000	,
aktüalite	,n	,00000000000000000000000000000000000000000	,0000011000	,
aktör	,n	,00000000000100000000000000000000000000000	,1010000000	,
akvaryum	,n	,00000000000000000000000000000000000000000	,0100000000	,
akıbet	,n	,00000000000000000000000100000000000000000	,0000011000	,
akıbet	,d	,1000000000000000000	,0000000000	,d1
akıl	,n	,00000010000000000000000000100000000000000	,0000011000	,
akım	,n	,00000000000000000000000000000000000000000	,0100001000	,
akın	,n	,00000010000000000000000000000000000000000	,0000000000	,
akşam	,n	,00000010000100001000000000000000000000000	,0000010000	,
akşam	,d	,1000000000000000000	,0000000000	,d1
al	,v7	,1110000000110000100	,0000000000	,v2 v4
al	,n a	,00000000000100000100000000000000000000000	,0000010000	,
alaca	,n a	,00000000000100001100000000000000000000000	,0000010000	,

alacak	,a	,00000000000000000000000000000001000	,0000000000	,
alaka	,n	,00000000000000000000000000000000000	,0000011000	,
alamet	,n	,00000000000000000000001000000000000	,0000010000	,
alan	,n	,00000000000000000000000000000000000	,0100000000	,
alarm	,n	,00000000000000000000000000000000000	,0000011000	,
alay	,n	,00000010010100000000000000000000000	,1010010000	,
aldan	,v4	,1000000000000000100	,0000000000	,v1 v4
aldat	,v4	,1000000000000000100	,0000000000	,v2
alet	,n	,00000000000000001000000100000000000	,0100000000	,
alev	,n	,00000000000000001000000000000000000	,0000001000	,
aleykümselam	,i	,1000000000000000000	,0000000000	,
alfabe	,n	,00000000000000000000000000000000000	,0100010000	,
algıla	,v5	,1010000000000000000	,0000000000	,v2
alkol	,n	,00001000001000010000000100000000000	,0100000000	,
alkış	,n	,00000010000000001101000000000000000	,0000001000	,
alt	,n a	,00000000001000010000000100000000000	,0100000000	,
altmış	,s2	,1000000000000000000	,0000000000	,
altı	,s0	,1000000000000000000	,0000000000	,
altın	,n a	,00000000000000000000000000000000000	,0100001000	,
alçak	,a	,00000000001000000100100000000000001000	,0000000000	,
alçal	,v8	,1000000000000000100	,0000000000	,v1
alçı	,n	,00000010000000001100000000000000000000	,0100001000	,
alıkoy	,v4	,1000000000000000000	,0000000000	,v2
alın	,n	,00000000000000000000000000100000000000	,0100000000	,
alın	,v13	,1000010000000000100	,0000000000	,v1 v4
alıngan	,a	,00000000001000000100100000000000001000	,0000000000	,
alıntı	,n	,00000000001000000000000000000000001000	,0000010000	,
alış	,v4	,1001000101100000100	,0000000000	,v1 v4
alışveriş	,n	,00000000000000000000000000000000000	,0000000000	,
ama	,c	,1000000000000000000	,0000000000	,
aman	,i	,1000000000000000000	,0000000000	,
amatör	,a n	,00000000001000000100000000000000000	,1010000000	,
amaç	,n	,00000000000000001100000000000000000	,0000010000	,
ambalaj	,n	,00000010000000001101000000000000000	,0100000000	,
ambar	,n	,00000010000000001000000000000000000	,0100000000	,
amblem	,n	,00000000000000000000000000000000000	,0100000000	,
ambulans	,n	,00000000000000000000000000000000000	,0100000000	,
amca	,n	,00000000001000000100000000000000000	,1010000000	,
ameliyat	,n	,01000000000000000000000100000000000	,0000000000	,
amir	,a n	,00000000001000000000000000000000000	,1010000000	,
ampul	,n	,00000000000000000000001000000000000	,0100000000	,
an	,v9	,1101100010000000001	,0000000000	,v2
an	,n	,00000000001000011110000000000000000	,0000010000	,
ana	,a n	,00000000001000000000000000000000000	,1010000000	,

anadili	,n	,00000000000000000000000000000100000000	,0000010000	,
anahtar	,n	,00000010000100000000000000000000000000	,0100000000	,
anane	,n	,00000000000000000010000000000000000000	,0000010000	,
anaokulu	,n	,00000000000000000000000000000100000000	,0100000000	,
anavatan	,n	,00000000000000000000000000000000000000	,1000000000	,
anayasa	,n	,00000010000000000010001000000000000000	,0000010000	,
ancak	,e	,1000000000000000000	,0000000000	,
ancak	,c d	,1000000000000000000	,0000000000	,d1
ani	,a	,00000000000000000000000000000000000011	,0000000000	,
ani	,d	,1000000000000000001	,0000000000	,d1
anket	,n	,00000010000100000000000010000000000000	,0000010000	,
anlam	,n	,00000001000000000000000000000000000000	,0000010000	,
anne	,n	,00000000001000000000000000000000000000	,1010000000	,
ansızın	,d	,1000000000000000000	,0000000000	,d1
ant	,n	,00000000000000000100000000000000000000	,0000010000	,
anten	,n	,00000000000000000000000000000000000000	,0100000000	,
antetli kağıt	,n	,00000000000000000000000000000000000000	,0100000000	,
antika	,n a	,00000010000100000010000000000000000000	,0100000000	,
antlaşma	,n	,00000001000000000000000000000000001000	,0000010000	,
anı	,n	,00000000000000000000000000000000000000	,0000010000	,
anıt	,n	,00000011010100100010000010000000001000	,0100000000	,
apartman	,n	,00000000000000000000000000000000000000	,0100000000	,
aptal	,a	,00000000001000000100000000000000000000	,0000000000	,
aptal	,i	,1000000000000000000	,0000000000	,
ara	,n	,00000010000000001101000000000000000000	,0000010000	,
ara	,v5	,1010000000000000000	,0000000000	,v2
araba	,n	,00000010000100000000000000000000000000	,0100000000	,
arabulucu	,n	,00000000001000000000000000000000000000	,1010000000	,
aralık	,n	,00000000000000000000000000000000000000	,0000010000	,

E.2. Representative Listing of the Translation Lexicon

A representative listing of the Translation lexicon, including its first 150 entries, is presented below in text form. Information for each entry is given in a single line. The fields in each entry are in the following order:

1. Azeri word
2. specific semantic features or concept activation information
3. index of the morphologically ambiguous root words entry

"ws" appearing as the Azeri word means that the corresponding Turkish word is word sense ambiguous. If the Azeri word field is empty that means the Azeri equivalent of the corresponding Turkish word is identical to it.

,66 123
artır ,
abdäst ,
,
böyük bacı ,
mühâsirä ,
abunä ,
äcâib ,
äcâib ,
äcälä ,
äcälä ,
äcämi ,
, ,2
, ,1
ac ,
kädärli ,
märhämätsiz ,
incit ,
, ,7
näzir et , ,9
,17 ,8

E.3. Morphologically Ambiguous Root Words Lexicon

Full listing of the morphologically ambiguous root words lexicon is presented below in text form. The fields in each entry are in the following order:

1. ambiguous Turkish word
2. flag to represent whether it is also word sense ambiguous or not
3. collocation info definition
4. concepts that the word is dependent on

An ambiguous word may have only collocation info definition or concept codes or both of them at the same time. 0 and 1 represent the false and true boolean values, respectively. "ci" means a collocation definition follows, whereas "co" means concept codes follow. In a collocation definition, collocations are separated with semicolons, and it ends with a period. The numeric concept codes are as follows:

Concept	Numeric code
army	1
medicine	2
agriculture	3
device	5
electricity	10
metal	13
grammar	15
navigation	17
mathematics	24
government	25
science	26
house	34
space	37
weapon	40
religion	42
education	43
law	44

acı 0 ci c(a) -;- ca(a)[+(dsav,dsnv)];- ca(n).
acı 0 ci ca(d) -.
aç 0 ci co18.
aç 0 ci - ca(n);ca(n) -[+dsav];- (doyurmak,beslemek);p2 -+pe.
aç 0 ci - ca(v).
açı 0 ci (radyanlık,derecelik) -;- ca(s) (derece,radyan)[+dsna];p2+ps -+ps.
ad 0 ci -a c94;co19.
ada 0 ci ca(a) -(n);-(n) ca(a)[+dsav];-ya (c118,çıkmak,ayak basmak);-yı (düzenlemek, imar etmek);co19.
co 17
ada 0 ci ca(n)+accu -ya.
adam 0 ci co2.
aday 0 ci 3+ps -ı;-a (oy vermek,rey vermek,güvenmek).
adım 0 ci - atmak;co3.
ağ 0 ci -a (takılmak,doldurmak,boşaltmak);co22.
ağa 0 ci co2;-cı 3;3 -cı;-ca böyle+te.
ağaç 0 ci co11;-a (çıkmak,tırmanmak).
ağla 0 ci co21.
ağrı 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
ağrı 0 ci ca(d) -.
ajan 0 ci co2.
ajans 0 co 47
ak 0 ci -lı [ca(a)] 2;benim -ım.
ak 0 ci ca(d) -.
akıl 0 ci 3+ps3s -ı.
akım 0 ci ca(a) -;c11 -ı.
al 0 ci co20.
al 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)];- ca(n).
alan 0 ci ca(a) -.
alın 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
alın 0 ci ca(d) -.
alt 0 ci ca(n)+geni -ı.
altı 0 ci - ca(n);- ca(s);- (tane,adet).
altın 0 co 13.

an 0 ci 3+accu -a;6t+accu -a;ca(d) -.
an 0 ci -a (yakışmak,uymak);ci -ı (hatırlamak,unutmamak).
ana 0 ci co2.
anı 0 ci - 3 için ca(a);3 için ca(a) -.
ara 0 ci c109 -ı;-lıktan (bakmak,girmek).
ara 0 ci - ca(d).
aralık 0 ci - ayı;ca(s) [yılı] -ı;ca(s) -.
art 0 ci co18.
art 0 ci co17.
as 0 ci ca(d) -ıl.
asıl 0 ci - ca(n);ca(n) -[+dsav].
asi 0 ci - [ca(a)] c130.
asil 0 ci -i ca(v).
asker 0 co 0
askeri 0 ci - c82;c82 -[+dsav].
aş 0 ci co14;- pişirmek.
aş 0 ci ca(n)+accu -;ca(d) -;ca(n)+accu -ın.
aşama 0 ci ca(a) -.
aşı 1 ci .
aşın 0 ci c127 -;-+dsva c127.
at 0 ci co5;-a (binmek,vurmak,koşmak,bağırmak);co22;sen -lasın.
at 0 ci o2+accu -;ca(n)+abla -.
ata 0 ci - (yadigarı,hatırası).
ata 0 ci o3+accu -.
atla 0 ci co21;ca(n)+abla -;ca(d) -.
atlas 0 ci senin -ın;-ı (oku,incele,ver,al).
az 0 ci ca(d) -a.
az 0 ci - ca(n);-a kanaat etmek.
az 0 ci - ca(v).
aza 0 ci ca(a) -;- ca(a)[+dsav];co3.
bağdaş 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
bağdaş 0 ci ca(d) -.
bağır 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
bağır 0 ci ca(d) -;c(n)+dati -.
bak 0 ci co20;ca(n)+dati -an.
bakan 0 ci ca(a) -;c69 -ı.
ban 0 ci ca(d) -;ca(n)+dati -.
bana 0 ci - ca(v).
barış 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
barış 0 ci ca(d) -.
bas 0 ci co23.
basın 0 co 62
beniz 0 ci ca(a) -.
benze 0 ci ca(d) -.

bez 0 ci ca(n)+abla -e.
bez 1 ci ca(a) -e.
beze 0 ci ca(n)+accu -.
bil 0 ci ca(d) -dik;ca(n)+accu -dik;c120+accu -.
bildik 0 ci - ca(n).
bile 0 ci ca(n)(n) -.
bile 0 ci c119+accu -.
bileği 0 co 5
bilek 0 ci co3.
bir 0 ci benim -im.
birim 0 ci (ca(s),kaç) -;c111 -i.
bit 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
bit 0 ci ca(d) -.
boğ 0 ci ca(d)-.
boğa 0 ci co5.
boya 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
boya 0 ci ca(d) -.
boz 0 ci co18;ci ca(d) -a;c121+accu -a.
boz 0 ci co17.
boza 0 ci co9.
böğür 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
böğür 0 ci ca(d) -.
bul 0 ci co17.
buluş 0 ci co18.
bur 0 ci co18;-uk 2.
buruk 0 ci - [ca(a)] tad;tadı [ca(a)] -.
buruş 0 ci co17.
büyü 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
büyü 0 ci ca(d) -.
çak 0 ci ca(d) -.
çakmak 0 ci ca(a) -.
çap 0 co 24
çapa 0 co 17
çekici 0 ci - ca(n).
çekiç 0 ci -i ca(v).
çıkar 0 ci ca(a) -;- ca(a)[+dsav].
çıkar 0 ci ca(d) -.
çıkış 0 ci ca(a) -;- ca(a)[+dsav].
çıkış 0 ci ca(d) -.
çıt 0 ci -a (dönmek,bakmak).
çıta 0 co 5
çiz 0 ci co18.
çizme 0 ci co17.
dal 1 ci co18.

dal 1 ci co17;a2 -a;ca(a) -a.
dala 0 ci 4[+dati] -.
damla 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
damla 0 ci ca(d) -.
de 0 ci 3+dati -;c99+accu -;ca(d) -sen.
değ 0 ci ca(d) -in;ca(n)+accu -in;co17;benim -im;ca(n)+dati -işim.
değin 0 ci d1+dati -.
değin 0 ci ca(d) -.
değiş 0 ci co18.
değişim 0 ci ca(a) -.
del 0 ci ca(d) -in;ca(n)+accu -in.
deli 0 ci senin -n;-n ca(v);ca(a) -.
dem 0 ci co24;-e [ca(n)] ca(v).
demin 0 ci - ca(v).
dene 0 ci ca(n)+accu -;ca(d) -ye.
deney 0 ci ca(a) -e.
der 0 ci c32+accu -.
dere 0 ci ca(a) -;- ca(a)[+dsav].
deri 0 ci senin -n;co3.
derin 0 ci - ca(n).
ders 0 ci ca(a) -e;-e ca(v).
desen 0 ci ca(a) -;-[ca(d)] ca(a) -[+dsav].
dev 0 ci -e doğru.
deve 0 ci co5.
devir 0 ci benim -im;-e kadar.
devir 0 ci ca(d) -.
devre 0 co 0
devrim 0 ci - yapmak;yapmak+dsva -.
dik 1 ci co18;co20;ca(n)+accu -en.
dik 0 ci - ca(n);ca(n) -.
diken 0 ci ca(a) -;4+geni -i.
dil 0 ci c102+accu -e.
dil 0 ci co3;co19.
dile 0 ci ca(n)+abla -;2(n) -;6t(n) -.
dilim 0 ci c102 -i;-i yemek;- c102.
din 0 ci co18.
din 0 ci co17;3+ps -i;a2 -i;c112 -i;-len [ca(d)] ca(v).
dinlen 0 ci ca(d) -.
diş 0 ci co3;1+ps3s -i.
dişi 0 ci - 1;1 -[+dsnav].
diz 0 ci co17;ca(d) -in;ca(n)+accu -in;siz -in.
diz 0 ci co18;co3.
dizi 1 ci .
dizin 0 ci c6+geni -i.

doğ 1 ci -an 3;-an bebek;ca(d) -.
doğa 0 ci senin -n;ca(a) -(n);-(n) ca(a)[dsav].
doğan 0 ci co6.
doğu 0 ci benim -m.
doğum 0 ci ca(a) -.
doku 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
doku 0 ci ca(d) -.
dol 0 ci ca(n)+loca -andır;c87 -;-muş [ca(a)] c87.
dolandır 0 ci 3+accu -.
dolmuş 0 ci ca(a) -;- [ca(d)] ca(v);- [bir] dolmuş.
don 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
don 0 ci ca(d) -.
dön 0 ci ca(s) - (yaptı,tamamladı).
dönüm 0 ci ca(s) -lük;ca(s) - c96;c96 ca(s) -.
duy 0 ci co18.
duy 0 ci co17.
co 10.
duyu 0 ci co3.
düş 1 ci ca(n)+abla -;ca(n)+dati -.
düş 0 ci 3 -ünde;3+ps -ünce;-ünde (görmek,konuşmak,söylemek,yaşamak).
düşün 0 ci ca(n)+accu -.
düşünce 0 ci c126 -;- c126[+dsnv].
edip 0 ci ca(a) -;- ca(a)+dsav.
efendi 0 ci benim -m;co2.
efendim 0 ci - [ca(d)] c94.
eğ 0 ci ca(d) -;ca(n)+accu -.
eğe 1 ci ca(a) -;- ca(a).
eğer 0 ci - ca(v)+dete.
ehli 0 ci - 4;4 -[+dsav].
ehil 0 ci 2+geni -.
ek 0 ci ca(d) -;5[+accu] -;ca(n)+dati -.
ek 0 ci senin -in;sen [ca(n)+loca] -sin.
co 15
ekin 0 ci co11.
eksen 0 ci co37;co24.
eksi 0 ci senin ca(s) -n.
ekşi 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)];- ca(n).
ekşi 0 ci ca(d) -.
el 0 ci -li [ca(a)] 1;ca(a) -(n).
ele 0 ci ca(d) -;ca(n)+accu -.
elli 0 ci - ca(s);- (tane,adet);- 2.
elma 0 ci co11;co14.
elmas 0 co 13.
emek 0 ci co22.

emekle 0 ci co21.
en 1 ci eğer -se.
ense 0 ci p2+ps -+ps;[p3] - [ca(d)] ca(v).
er 0 ci ca(d) -.
er 1 ci co17.
er 0 ci - ca(v).
eri 0 ci c84 [ca(d)] -+.
eriş 0 ci co18.
erişte 0 ci ca(a) -;co14.
ermiş 0 ci co2.
es 0 ci -en c98.
esen 0 ci - kalmak.
eski 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)];- ca(n).
eski 0 ci ca(d) -.
eş 1 ci co24.
eş 0 ci ca(d) -.
eşin 0 ci ca(d) -;ca(n)+accu -.
et 0 ci d1 -ken.
et 0 ci yapıp edip;ca(n) -.
etken 0 ci (önemli,asıl,gerçek,tek,ca(s)+dsna2) -.
fil 0 ci (biz,hepimiz,topumuz) -iz;co5;-e doğru.
file 0 ci - (almak,satmak,doldurmak).
filiz 0 ci ca(a)-;co11.
geç 0 ci ca(n)+abla -e.
geç 0 ci - ca(v).
geç 0 ci - ca(n);ca(n) -[+dsav].
geçe 0 ci (ca(s),biraz,az) [c36] -.
geçe 0 ci - ca(v).
geçende 0 co 0
gerek 0 ci ca(n)(n) -sin.
gerek 0 ci sen -sin.
gereksin 0 ci ca(n)+dati -.
ger 0 ci ca(n)+accu [ca(d)] -in.
geri 0 ci senin -in.
geri 0 ci - ca(v).
gerin 0 ci 3 -.
gir 0 ci 1 [ca(d)] -di;ca(n)+dati -işim.
girdi 0 ci ca(s) c89+dsna6 -.
girişim 0 ci ca(a) -;- ca(a).
göç 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
göç 0 ci ca(d) -.
gönder 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
gönder 0 ci ca(d) -.
göz 0 ci -+loca [ca(a)] ca(n) (var,yok).

gözde 0 ci - [ca(d)] ca(v).
gül 0 ci -la (beraber,birlikte).
gül 0 ci ca(d) -.
gülle 0 ci - (atmak,fırlatmak,savurmak).
güreş 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
güreş 0 ci ca(d) -.
güven 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
güven 0 ci ca(d) -.
hac 0 ci c36+pron -ın;-ın [ca(a)] ca(n)+accu.
hacı 0 ci senin -n.
hal 0 ci -e (mal,sebze,meyve gelmek).
hala 0 ci co2;3+ps -sı.
halas 0 ci 6t+ps -ı;6t 3+ps -ı.
hale 0 ci co24;- (oluşmak,görmek).
halen 0 ci - [ca(n)] ca(v).
halk 0 ci -a c94;c94+dsva -a.
halka 0 ci - (takmak,geçirmek).
han 1 ci co19.
hanım 0 ci 3+geni -.
hırs 0 ci (biz,hepimiz,topumuz) -ız.
hırsız 0 ci ca(a) -;- ca(n)+dati girmek;- (yakalanmak,çalmak).
has 0 ci -ta ca(n)+dsnn7 var;ca(p) -ta ca(p);benim -ım;p2 -sa+ps;-sa (ca(p),ca(n)).
hassa 0 ci p2+ps -+ps;p3 [ca(d)] ca(v).
hassas 0 ci -ı [daha] ca(a);ca(n)+abla [daha] -ı.
hasım 0 ci ca(a) -.
hasret 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
hasret 0 ci ca(d) -.
hasta 0 co 2
hat 0 ci -ta+.
hatta 0 ci ca(v) - [ca(d)] ca(v);ca(n) - ca(n);ca(a) - ca(a).
hava 0 ci co24.
havan 0 ci -da (dövmek,tavlamak).
iç 0 ci co17;senin -in ca(a)[+dsav].
iç 1 ci co18;ca(d) -in;ca(n)+accu -in;ca(d) -işleri;c22 -işleri.
için 0 ci ca(n)+geni -.
içişleri 0 ci - (bakanı,müsteşarı,bakanlığı,memuru,bürokratı,binası).
il 0 ci co19;o4 -;ca(a) -e.
ile 0 ci ca(n) - ca(n).
ilim 0 ci c95 ca(a) -.
ilk 0 ci -e ca(v)+dsvv15.
ilk 0 ci - ca(v).
ilke 0 ci - (edinmek,kabul etmek).
in 0 ci co17;co22.

in 0 ci co18.
inle 0 ci co21.
inme 0 ci co17;c134+dati - inmek;
is 0 ci ca(a) -(n);benim -im.
isim 0 ci ca(a) -;- ca(a)[+dsav].
iste 0 ci ca(d) -.
iş 0 ci - [ca(d)] ca(v).
işte 0 ci - a2 ca(n).
it 0 ci co18.
it 0 ci co17.
kaba 0 ci co15.
kaç 0 ci ca(d) -.
kaç 0 ci ca(a) -;- ca(n).
kaçar 0 ci - kaçar;- (tane,adet,ca(n)).
kal 0 ci ca(d) -;ca(n)+loca -;co23.
kala 0 ci (ca(s),yarım) -;c36 -;(ca(s),yarım) c36 -ya;çeyrek -ya.
kalas 0 ci senin -ın;ca(a) -ın;-ın ca(v).
kalay 0 co c13
kale 0 ci co19.
co 1
kalem 0 ci -le (yazmak,çizmek,boyamak);-(n) ca(v)+pe.
kalın 1 ci - ca(n).
kalıp 0 ci ca(a) -.
kalk 0 ci co20;ca(n)+abla -ın.
kalkan 1 ci .
kalkın 0 ci (o3,o2) -.
kan 0 ci ca(n)+dati -a.
kan 0 ci ca(a) -(n);- (akmak,pıhtılaşmak,durmak);(a,b,0,ab) gurubu -;rh (pozitif,negatif) -.
kana 0 ci c113+accu -;ca(d) -t.
kanarya 0 ci co6;- (ötmek,şakımak).
kanat 0 ci - takmak;ca(a) -;- [ca(d)] ca(a)[+dsav].
kanı 0 ci 6t+loca+pron -;hakkındaki -.
kap 1 ci -a (dökmek,koymak,boşaltmak);-lan (beraber,birlikte);-lan ca(n) ca(v);ca(a) [ca(a)] -sa;-tan (birşey olmaz,hayır gelmez).
kap 0 ci ca(d) -;ca(n)+accu -;co20;1 ca(n)+accu -sa;ci ca(n)+accu -sam;ben -sam;ca(d) -sam;eğer -sa(n).
kapalı 0 ci - ca(n).
kapan 0 ci - kurmak;-a (takılmak,yakalanmak);co10.
kapı 0 ci senin -n;ca(a) -n.
kaplan 0 ci co5.
kapsa 0 ci 6t+accu -;
kapsam 0 ci ca(a) -;- ca(a)[+dsav].
kaptan 0 ci co2.

kar 0 ci ca(d) -a;ca(n)+accu -a;ca(d) -ın;siz -ın;ca(n)+accu -ın;ca(n)+accu -ınca;ca(d) -ınca;ca(d) -ışım;ca(n)+accu -ışım.
kar 0 ci -a (bakmak,basmak,dokunmak);-ı (süpürmek,tuzlamak);-ın yağmak.
kara 1 ci -ca [ca(a)] ca(n);ca(n) [ca(d)] -ca[+dsav].
karaca 0 ci co18.
karar 0 ci ca(a) -;- ca(a)[+dsnv].
karar 0 ci ca(n)(n) -.
karı 0 ci 3+ps -ı;co2;senin -n;senin -nca.
karın 0 ci co3.
karınca 0 ci ca(a) -;- ca(a)[+dsnv].
karış 0 ci ca(n)+dati -.
karış 0 ci benim -ım;co3;ca(s) -.
karışım 0 ci ca(a) -;- ca(a)[+dsnv].
kas 0 ci ca(d) -a;ca(n)+accu -a;-ık ca(n).
kas 0 ci co3.
kasa 0 co 5
kasaba 0 ci c115 -;- c115.
kasap 0 ci co2.
kasık 0 ci co3.
kasım 0 ci - ayı;-da ca(v);ca(s) [yılı] -ı;ca(s) -.
kat 0 ci ca(s)+dssa2 -;(zemin,bodrum,giriş,ilk,alt,üst) -;3+ps -ı.
kat 0 ci ca(d) -.
katı 0 ci - [ca(a)] c83;c83 [ca(d)] - [+dsav].
katı 0 ci - ca(v).
kay 0 ci co23;ca(d) -ıp;ca(n)+abla -ıp;ca(n)+abla -;ca(n)+loca -;ca(d) -;ca(n)+abla -;-ık ca(n).
kaya 0 ci ca(a) -.
kayık 0 co 17
kayın 0 ci co11.
kayıp 0 ci ca(a) -;c36+abla beri -;c36+eecg -.
kayış 0 ci co4.
kaymak 0 ci ca(a) -.
kaz 0 ci co5.
kaz 0 ci -an 3.
kaza 0 ci senin -n;c175 -;trafik -;-n o4.
kazan 0 ci co15.
kazan 0 ci ca(n)(n) -.
kel 0 ci benim -ime;-ime (dokunmak,laf söylemek);-le (beraber,birlikte);-le uğraşmak;kafa+ -.
kelle 0 ci - (pişirmek,yemek,satmak);(pişirilen,yenilen,satılan) -;(yediği,sattığı,pişirdiği)+ps.
kelime 0 co 15
kes 0 ci ca(d) -;ca(n)+accu -;(4,4) -.
kese 0 ci ca(a) -;- ca(a)[+dsnv].

kesin 0 ci - ca(n);- ca(v).
kesim 0 ci ca(s)+dssa2 -;c75 -.
kıl 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
kıl 0 ci ca(d) -.
kın 0 ci c123+accu -a (koymak,geçirmek).
kına 0 ci -sürmek;baş+ - koymak.
kına 0 ci ca(d) -.
kır 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)];- ca(n).
kır 0 ci ca(d) -.
kıs 0 ci ca(d) -;keşke -sa.
kısa 0 ci -sı [daha] ca(a);ca(n)+abla [daha] -sı.
kısa 0 ci - ca(v).
kısas 0 ci -ı uygulamak;-ını (çözmek,unutmak).
kıssa 0 ci ca(a) -;- ca(a).
kıy 0 ci co18.
kıyma 0 ci co17.
kız 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)];- ca(n).
kız 0 ci ca(d) -.
koca 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)];- ca(n).
koca 0 ci ca(d) -.
kon 0 ci co17.
konu 0 ci benim -um.
konum 0 ci ca(a) -;ca(n)+geni -u;3+ps -u.
konuş 0 ci co18.
kor 0 ci ateşin -;- ateş;a2 - [ca(d)] v1.
koru 0 ci -da c85.
koru 0 ci ca(d) -.
koş 0 ci 4 ca(n)+dati -ul;benim -um.
koşul 0 ci ca(a) -.
koşum 0 ci - (takımları,aletleri).
kot 0 ci -a (bakmak,sürünmek,dökülmek).
kota 0 ci - (uygulamak,koymak).
kov 0 ci ca(n)+abla -a;ca(d) -a;-an ca(n).
kova 0 ci ca(a) -;- ca(a);senin -n.
kovan 0 ci arı -ı;- balı;-da (bal,arı) (var,yok).
koy 0 ci senin -un.
co 27
koy 0 ci ca(n)+dati -un;ca(d) -un.
koyu 0 ci - [renk] [ca(a)] ca(n);rengi [ca(a)] -.
koyun 0 ci co5.
koz 0 ci -a güvenmek.
koza 0 ci - (toplamak,dökülmek,olmak,olgunlaşmak).
kum 0 ci -a (yatmak,uzanmak,boşaltmak).
kuma 0 ci - (gelmek,getirmek,olmak).

kur 0 ci c114+accu -a;ca(d) -a;ca(d) -maya.
kura 0 ci - çekmek;çekilen -;ca(a) -.
kurmay 0 co 1
kuru 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)];- ca(n).
kuru 0 ci ca(d) -.
küf 0 ci -e (bulanmak,bulaşmak).
küfe 0 co 5
küme 0 co 24
kümes 0 ci 4 -i;- 4.
küp 0 ci -e (koymak,doldurmak,boşaltmak,vurmak).
küpe 0 ci - (takmak,almak,satmak).
küs 0 ci - ca(a)[+(dsav,dsnv)];- ca(n).
küs 0 ci ca(d) -.
laf 0 ci (biz,hepimiz,topumuz) -ız.
lafız 0 co 0
lisan 0 ci c21 -sa;-sa (onu,bunu) halletmek.
lisans 1 ci .
mart 0 ci ca(s) yılı -ı;- ayı.
martı 0 ci co6.
memnu 0 ci co24.
memnun 0 ci - [ca(a)] -.
mera 0 ci co19;-mda otlatmak.
meram 0 ci -ı c94.
nal 0 ci senin -ın.
nalın 0 ci - (almak,takmak,giymek,satmak).
ne 0 ci -ye ca(v) mood(int).
ne 0 ci -ye ca(v) mood(int).
neden 0 ci eğer -se;ca(n) [bir] -.
nedense 0 ci her -;- sent.
ney 0 ci eğer -se;co12.
neyse 0 ci - ca(v).
oda 0 ci ca(a) -;- ca(a)[+dsav].
o 0 ci co2;eğer -ysa;ca(v)+dsva -ysa;ben -yum.
ol 0 ci co17;ca(n)+dati -uşum;(öyle,böyle) -uşum.
oluş 0 ci co18.
oluşum 1 ci ca(a) -;- ca(a)[+dsav].
on 0 ci -ar onar;-ar (tane,adet);-ar ca(n).
onar 0 ci ca(d) -;ca(n)+accu -.
ora 0 ci senin -na;-na burana.
oran 0 ci a2 -a;ca(a) -a.
orta 0 ci benim -ma.
ortam 0 ci a2 -[a];ca(a) -[a].
ot 0 ci (biz,hepimiz,topumuz) -uz.
otuz 0 ci - ca(n);- ca(s);- (tane,adet).

ov 0 ci ca(d) -.
ova 0 ci ca(a) -;- ca(a).
oy 0 ci ca(n)+accu -(un,sa);ca(d) -.
oy 0 ci senin -un;-a (bağlanmak,ümit bağlamak);benim -um;-um 3+dati;eğer -sa.
oya 0 ci - (örmek,dikmek,işlemek).
oymak 0 ci ca(a) -.
oysa 0 ci ca(v)+dsva -.
oyun 1 ci ca(a) -.
öd 0 ci co24;-ün (kopmak,patlamak).
ödün 0 ci - (vermek,almak).
öğüt 0 ci ca(a) -;- ca(a)[+dsav].
öğüt 0 ci ca(d) -.
örs 0 ci -e ca(v);ca(a) -e.
ör 0 ci c31 -se;ca(d) -se.
öt 0 ci ca(d) -;c105 -.
öte 0 ci ca(n)+abla -.
özen 0 ci ca(a) -;- ca(a)[+dsav].
özen 0 ci ca(d) -.
pas 0 ci -tan (batmak,geçilmez olmak)+(pe1s,pe1p,pe2s,pe2p,pe3p).
pasta 0 ci senin -n;co14.
peki 0 ci -yi [hiç] c94.
pekiyi 0 ci - (almak,vermek);not+ps -.
peş 0 ci co24;-inde ca(n) (var,yok).
peşin 0 ci - para;- [ca(a)] ca(n).
peşin 0 ci - ca(v).
pir 0 ci co2;senin -ince.
pir 0 ci - ca(v).
pire 0 ci co5.
pirinç 1 ci a2 -e.
rahibe 0 ci -o8.
rahip 0 ci - o7 -e.
rakı 0 ci co19;co9.
rakım 0 ci (o3,o4,o5,o6)+geni -ı;ca(s) -lık.
rehin 0 ci -e bakmak;ca(n)+accu -e ca(v).
rehine 0 ci - (olmak,tutmak,saklamak).
resim 0 ci 3+ps -;- +ps.
resmi 0 ci - [ca(a)] c81;c81 -[+dsav].
saç 0 ci co18.
saç 0 ci co17.
saçma 0 ci co17.
sağ 0 ci co22.
sağ 0 ci ca(d) -.
sağla 0 ci co21;ca(n)+accu -ma;ca(d) -ma.

sağlam 0 ci - ca(n)+dati;ca(a) -a.
sağlam 0 ci - ca(v).
saha 0 ci co24;-nda oynamak.
sahan 0 ci - ca(n) (pişirmek,yemek).
sal 0 ci co18;ca(n)+accu -ın.
sal 0 ci co17;ca(a) -;co22;3+ps -ı[+dsnv];- v1;senin -ın.
saldır 0 ci ca(d) -;ca(n)+dati -.
salı 0 ci - gün(ü,leri);- v2.
salın 0 ci ca(n)(n) -.
salla 0 ci co21.
sap 0 ci ca(n)+abla -;co20;ca(d) -ana;c116+abla -ana.
sap 1 ci ca(a) -;co22;c51+loca+pron -ta;5+loca+pron -ta.
sapa 0 ci - ca(n)ca(n) [ca(a)] -.
sapan 0 ci co10;-a taş (koymak,ip bağlamak);ca(a) -a;c117+abla -a.
sapla 0 ci co21.
sapta 0 ci ca(n)+accu [ca(d)] -.
sara 0 co 2
saray 0 ci -a c118.
savaş 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
savaş 0 ci ca(d) -.
say 0 ci ca(n)+accu -;ca(d) -ın.
sayı 0 ci senin -n.
sayın 0 ci -3.
sayman 0 ci co2.
seci 0 ci (ahenkli,uyumlu,kafiyeli) [bir] -ye.
seciye 0 ci co1.
sek 0 ci ca(n)+abla -sen;ca(d) -sen.
seksen 0 ci - ca(s);- tane;- ca(n).
sel 0 ci -e (tutulmak,kapılmak,yakalanmak,engel olmak).
sele 0 co 5
semer 0 ci -e (oturmak,binmek,kurulmak).
semere 0 ci - (vermek,almak).
ser 0 ci ca(n)+dati -;ca(n)+accu -.
serin 0 ci - ca(n);ca(n) -[+dsav].
sersem 0 ci co1.
sığ 0 ci - ca(n);ca(a) -.
sığ 0 ci ca(d) -.
sıva 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)].
sıva 0 ci ca(d) -.
sivil 0 ci -ce [ca(a)] c86;c86 -ce[+dsav].
sivilce 0 co 2
sol 0 ci ca(a) -;2+ps -u.
sol 0 ci ca(d) -.
solu 0 ci 3 [d2+abla] [d1] -;ca(d) -.

som 0 ci - ca(n);ca(n)+accu -.
somun 0 co 5
sor 0 ci co23.
sorun 0 ci - (yapmak,oluşturmak).
soy 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)];- ca(n).
soy 0 ci ca(d) -.
sulta 0 ci co24.
sultan 0 ci co2.
sür 0 ci c106+accu -.
süre 0 ci ca(a) -.
sürü 0 ci ca(d) -.
sürü 0 ci benim -m;-mü (otlatmak,beslemek).
sürüm 0 ci c123+loca -;c12+geni -ü;- c123.
şiş 0 ci -e (takmak,geçirmek);co1.
şiş 0 ci ca(d) -.
şişe 0 ci - c22;co9;- kırmak.
tak 0 ci ca(d) -;ca(n)+dati -.
tak 0 ci -la (beraber,birlikte);-la (süslemek,donatmak).
takım 1 ci ca(a) -;- ca(a)[+dsav].
takla 0 ci - atmak;(atılan,attığ+ps) -.
tan 0 ci (ağaran,doğan) -;- (ağarmak,doğmak).
tandır 0 ci - (yanmak,pişirmek).
tanı 0 ci ca(d) -;ca(n)+accu -.
tanım 0 ci a2 -la;ca(a) -la.
tanımla 0 ci ca(n)+accu -;ca(d) -.
tarif 0 ci -e bakmak.
tarife 0 ci - (almak,sormak).
tarih 0 ci -çe ca(n) ca(v).
tarihçe 0 ci - (yazmak,okumak).
tart 0 ci co17.
tartış 0 ci co18.
taş 0 ci ca(a) -;- ca(a)[+dsav].
taş 0 ci ca(d) -.
tat 0 ci ca(a) -;- ca(a)[+dsav].
tat 0 ci ca(d) -.
tavla 0 ci ca(a) -;- ca(a)[+dsav].
tavla 0 ci ca(d) -.
tel 0 ci senin -in;-siz ca(v)+negp.
co 10
telin 0 ci - etmek.
telsiz 0 co 5
ter 0 ci co19;ci -imi (silmek,yıkamak,kurulamak);-im (soğumak,kurumak).
terim 0 co 24
tos 0 ci co24;-un [ca(d)] (hırpalamak,acıtmak,öldürmek).

tosun 0 ci co5.
töre 0 ci co24;-ne bağlı+dsav.
tören 0 ci - (yapmak,kutlamak).
tür 0 ci ca(a) -.
türe 0 ci ca(d) -;ca(n)+abla -.
türedi 0 ci - [ca(a)] ca(n).
tüt 0 ci co23.
tütün 0 ci - (içmek,sarmak);co11.
tüy 0 ci ca(a) -;- ca(a)[+dsav].
tüy 0 ci ca(d) -.
uç 0 ci ca(a) -;- ca(a)[+dsav].
uç 0 ci ca(d) -.
ufal 0 ci ca(n) -.
ufala 0 ci ca(n)+accu -.
uğraş 0 ci ca(a) -;- ca(a)[+dsav].
uğraş 0 ci ca(d) -.
ulu 0 ci - ca(n);ca(a) -.
ulu 0 ci ca(d) -.
uy 0 ci -an ca(n);ca(n)+dati [ca(d)] -;ca(a)+dati -.
uyan 0 ci d1 -.
uyar 0 ci 1+accu -.
uydu 0 co 37
var 0 ci ca(a) -;- ca(a)[+(dsav,dsnv)];- ca(n).
var 0 ci ca(d) -.
veri 0 ci co19
verim 0 ci c93+geni -i;-(li,siz) c93.
vezin 0 ci -e (uymak,dikkat etmek).
vezne 0 ci - memuru;- [ca(a)] (meşgul,dolu,yoğun,kalabalık);(meşgul,dolu,yoğun, kalabalık) -.
yağ 0 ci ca(a) -;- ca(a)[+dsav].
yağ 0 ci co18.
yağma 0 ci co17.
yak 0 ci ca(n)+accu -;ca(n)+accu -ın;ca(d) -;co17.
yaka 1 ci ca(a) -.
yakar 0 ci ca(n)+dati -.
yakın 0 ci - ca(n);ca(n) -[+dsav].
yakın 0 ci - ca(v).
yakın 0 ci ca(d) -;-[+dsva] ca(n).
yakış 0 ci co18.
yama 0 ci ca(d) -;c31+accu -.
yama 0 ci ca(a) -;c31+loca+pron -.
yan 0 ci ca(a) -;- ca(n).
yan 0 ci ca(d) -.
yan 0 ci - ca(v).

yap 0 ci co17.
yapış 0 ci co18.
yar 0 ci ca(n)+accu -;-an ca(n).
yar 0 ci c125 -;-[(ım,ın)]a c118.
yara 0 ci ca(n)+dati -.
yara 0 ci ölümcül -;1+loca+pron -;senin -n.
yaran 0 ci 3+dati -.
yarar 0 ci - (görmek,ummak,beklemek);ca(a) -.
yarı 0 ci - ca(n);- ca(n)+loca.
yarı 0 ci - ca(v).
yarım 0 ci - ca(n);- c36.
yarın 0 ci - (sabah,öğle,akşam,gece);- ca(v)+(pe1s,pe1p,pe3s,pe3p,pe2p).
yarın 0 ci - ca(v).
yarış 0 ci ca(a) -;- ca(a)[+dsav].
yarış 0 ci ca(d) -.
yasak 0 ci co22.
yasakla 0 ci co21.
yaşa 0 ci ca(d) -sın;1 -sın.
yaşa 0 ci - (de,diye bağır).
yaşasın 0 ci mood(exc).
yat 0 ci co17.
yatış 0 ci co18.
yay 0 ci -a ok takmak;senin -ın;co10;-la ok c122.
yay 0 ci ca(n)+accu -;-an 3;ca(n)+dati -;-ık ca(n);ca(n) -ık.
yaya 0 ci - (yürümek,koşmak).
yayan 0 ci - 3;3 -.
yayan 0 ci - ca(v).
yayık 0 ci ca(a) -;-+cases ca(v).
yayın 0 ci - yayınlamak.
yayla 0 ci c175 -;- c175[+dsav];c118+dsav -.
yaz 0 ci co18.
yaz 0 ci co23;co17;-ı ca(n)+loca geçirmek;(sıcak,güneşli,yağmurlu,soğuk,ılık)
-.
yazı 0 ci - yazmak;3+ps -;c124 -;- c124[+dsav].
yazın 0 ci - ca(v).
ye 0 ci c23[+accu] [ca(d)] -.
yedi 0 ci - ca(n);- ca(s);- (tane,adet).
yemiş 0 ci c041.
yen 0 ci co17;ca(s) -;(kaç,nekadar) -.
yen 0 ci co18.
yer 0 ci ca(a) -;- ca(a)[+dsav].
yer 0 ci ca(d) -.
yet 0 ci co17;ca(n)+dati [ca(d)] -.
yetiş 0 ci co18.

yetmiş 0 ci - ca(n);- ca(s);- (tane,adet).
yık 0 ci c108+accu -.
yıka 0 ci c107+accu -.
yıl 0 ci co20.
yıl 0 ci co17.
yılan 0 ci co7.
yılış 0 ci co18.
yok 0 ci 3 [ca(n)+loca] -sun.
yok 0 ci - ca(v).
yoksun 0 ci - ca(n);ca(n) -[+dsav].
yol 0 ci ca(a) -;- ca(a)[+dsav].
yol 0 ci ca(d) -.
yüz 1 ci .
yüz 1 ci .
yüzde 0 ci co24.
yüzden 0 ci p3 -.
zarf 0 ci senin -ın;(mektubun,kartın) -ında;-ında (pul,mühür,adres,damga) (yok,var,olmak).
zarfında 0 ci c36 -;süre -.

E.4. Word Sense Ambiguous Root Words Lexicon

Full listing of the word sense ambiguous root words lexicon is presented below in text form. For each entry the ambiguous Turkish word and the number of its word senses are given in the first line. The related information for each word sense is stored in the next lines. The fields in each word sense are in the following order:

1. the equivalent Azeri word
2. collocation info definition or the concepts that the word is dependent on

A word sense may have only collocation info definition or concept codes or both of them at the same time. If it utilizes both of them, then the word sense is stored with each of them one by one. 0 and 1 represent the false and true boolean values, respectively. "ci" means a collocation definition follows, whereas "co" means concept codes follow. In a collocation definition, collocations are separated by semicolons, and it ends with a period. The same numeric codes presented in Section E.3 is used for concepts.

```
alay                2
alay                co  1
lağ                 ci ca(n)+iles -(n) (etmek, geçmek);-a(n) (almak, vurmak);-la(n)
(bakmak,süzmek);-cı.
aşı                 2
peyvänd             co  2
calag               co  3
ayar                2
ayar                ci ca(s) - altın;altının -ı.
däqiqlik            ci 4+geni -ı.
bağlı               2
bağlı               ci c7+iles -;2 2+dati [ca(a)] -[+dsav].
asılı               ci 6t 6t+dati [ca(a)] -[+dsav];1 1+dati [ca(a)] -[+dsav].
baskı               2
näşr                ci 6+geni [(ca(s)+dssa2, kaçıncı)] -sı;ca(s) -.
täzyiq              ci (3, 2+dati) (yapılan, üzerindeki) -;(3, 2+dati) [ca(a)] - yapmak.
başıboş             2
```

avaraci	co 1
yiyäsiz	ci - [ca(a)] 4+;4 [ca(d)] -.
ben	2
xal	co 2
men	ci -[im] ca(a)+ps1s;- ca(v)+pe1s.
bez	2
bäz	co 2
bez	ci -+iles ca(v);- dikmek.
cereyan	3
cäräyän	co 10
cäräyän	ci c11 -ı.
hava axını	ci - yapmak;hava -ı.
cetvel	2
xätkeş	co 5
cädväl	ci - (hazırlamak, çıkarmak).
cevher	2
cävhär	co 13
cävâhir	ci co1.
cilt	2
däri	ci 3+geni -i.
cild	ci c12+geni -i;-li [ca(a)] c12.
cümle	2
cümlä	co 15
bütün	ci - ca(n)+plrp.
çağdaş	2
hämäsr	ci .
çağdaş	ci .
çapa	2
külüng	co 3
lövbär	co 17
çat	2
baş-başa goy	ci ca(n)+accu [ca(d)] -.
hücum et	ci 3+dati [ca(d)] -.
çatal	2
çängäl	ci -la yemek;-lı (yıkamak, temizlemek, silmek).
ayrıc	co 0
çekim	2
täsriflänmä	ci fiil[(in, inin)] -i;ca(v)+dsvv17+geni -i.
çäkmä	co 0
çerçöp	2
çör-çöp	ci -+accu yakmak.
zir-zibil	co 0
çevir	3
tärcümä et	ci yazıyı, metni -;o6+dati -;o6+abla -.

mühasiräyä al	co 1
çevirmek	ci 2+accu -.
çırp	2
çarp	ci (c16, c79)+accu -.
äl çal	ci el+ps+accu [ca(d)] -.
çiğne	2
çiynä	ci c23 [ca(d)] -.
çığna	ci c63 [ca(d)] -;c72+iles -.
daire	3
dairä	co 24
idarä	co 25
mänzil	co 64
dal	2
budag	co 3
ixtisas	co 26
dal	3
uyguya dal	ci uykuya -.
dal	ci (düşünceye, hayale) -.
cum	ci c23+dati -.
darbe	2
çevriliş	co 25
zärbä	ci - (aldı, yedi);(aldığı, yediği) [ca(a)] -.
dava	2
dava	co 44
ideal	ci (ulvi, yüce) [ca(a)] -;- için 3+accu feda etmek.
dayak	2
kötäk	ci -(n) (atmak, vurmak, yemek).
dayag	ci -(ı, la) [ca(d)] (koymak, yerleştirmek, sıkıştırmak).
dayı	2
dayı	ci o1+ps3s -sı;-ım;-ın.
arxa	ci c28+geni -sı.
değerlendir	2
räy yaz	ci c29+accu -.
däyärländir	ci .
deneme	2
esse	co 30
täcrübä	ci - [ca(d)] yapmak.
denk	2
yük	ci -i (yüklemek, taşımak, kaldırmak, sırtlanmak);ca(a) -.
tän	ci ca(s)+accu+loca -;hepsi -;- ca(n)+accu;
derece	2
termometr	co 5
däräcä	ci ca(s) -;kaç -.
derman	2

därman	co 2
güc	ci .
dik	3
äk	ci 5 -;-+dsva 5.
tik	ci c31 -;-+dsva c31.
basdırmak	ci (direk, direği) [ca(d)] -.
divan	2
divan	ci - (okumak, yazmak).
taxt	ci -a (oturmak, yatmak, uzanmak);-dan kalkmak.
diye	2
deyä	ci (ca(r), ca(a)) -.
üçün	ci ca(v) -.
diyet	2
pähriz	co 2
girov	ci -ini (ödemek, almak, vermek).
dizi	2
seriya	co 24
düzüm	ci - [ca(a)] ca(n).
doğ	2
doğ	ci c33[+geni] -+.
doğul	co 2
don	2
trusu	ci -+ps;co4.
don	ci - (yapmak, yağmak).
dosya	2
papka	co 45
şäxsi iş	co 44
duvar	2
divar	ci ev+ps -ı;ev[in] -ı.
hasar	ci c35+ps -ı;c35[+geni]-ı.
düş	2
düş	ci .
yıxıl	ci .
düzen	2
sistem	ci .
tärtib	ci .
efendim	3
bağışlayın	ci .
bäli	ci .
bäli beyim	ci .
eğe	2
gabırğa	ci - kemiği.
yeyä	co 5
eğitim	2

pedagogika	co 26
tärbiyä	ci - vermek;-(siz, li) [ca(a)] 3;3+geni -i.
eğlen	2
lâğa goy	ci 3+iles -.
äylän	co 0
eğreti	2
ötäri	ci - [ca(a)] c36.
pozug	ci - [ca(a)] 2.
ek	2
şäkilçi	co 15
goşma	ci .
el	2
äl	co 7
el	co 25
emsal	2
ämsalci	3 3+geni -.
nümunä	ci ca(n) [ca(a)] ca(n) -.
en	2
än	ci - ca(a).
en	ci -+.
er	3
äsgär	co 1
kişi	ci .
märd	ci .
esne	2
gärilib yığıl	ci 2 [ca(d)] -+.
äsnä	ci 1 [ca(d)] -+.
eş	2
yoldaş	ci 3+ps -i.
tay	ci 2+ps3s -i.
etek	2
tuman	co 16
ätäk	ci c73+geni -i.
fail	2
fail	co 15
mügäsir	ci (eylem, cinayet, suç, olay)+geni -i.
fen	2
täbiät fänläri	co 26
texnika	ci .
fener	2
mayak	ci deniz -i;-ci.
fänär	co 0
fıkırda	2
pıggılda	ci c74 [ca(d)] -.

gımılda	ci 3 [ca(d)] -.
fikra	2
oçerk	co 30
lätifä	ci - anlatmak;-ya [ca(d)] gülmek;(komik, gülünç) [ca(a)] -.
fiş	3
çängäl	co 10
çek	ci ca(s)+dsnn7 -;- vermek.
kartoçka	ci -e (işlemek, kaydetmek).
fişek	2
patron	co 40
fişäng	ci havai fişek.
garip	2
qärib	ci .
qäribä	ci .
gebe	2
hamilä	ci - [ca(a)] 3;3 [ca(d)] - [+dsav].
boğaz	ci - [ca(a)] 4+;4 [ca(d)] -[+dsav].
göbek	2
garın	ci .
göbäk	co 0
gövde	2
äsas	co 15
gövdä	ci co3;5[+ps] -+ps.
gözle	2
gözlä	ci .
müşahidä et	ci .
güç	2
güc	ci -(ps, loca, abla, geni, dati, accu, lü, süz);- kullanmak.
çätin	ci -lük;- [ca(a)] ca(n);ca(d) -.
hak	2
hagg	ci -ı tessis etmek.
hügug	ci 3+geni -+ps;-+ps+accu vermek.
han	2
kârvansaray	co 34
xan	ci co2.
harç	2
xärc	ci - (toplamak, vermek, ödemek).
mähllul	ci - (karıştırmak, hazırlamak, dökmek).
hava	2
ähval-ruhiyyä	ci -+ps.
hava	co 0
havale	2
hävälä	co 2
köçürmä	ci - (göndermek, çekmek);- almak;- ca(d)+dati ulaşmak.

hayır 2
xeyr ci - [ca(d)] cevabı;negsen -.
xeyir ci - yapmak;- dua etmek.
hesaplı 2
hesablanmış ci - [ca(a)] c22;c22 [ca(a)] -.
tädbirli ci - [ca(a)] 3;3 [ca(a)] -+te.
hoca 2
xoca co 42
müällim co 43
horla 2
xorulda ci 3 [ca(d)] -;uyku+loca -.
xorla ci 3+accu [ca(d)] -;3 [ca(d)] -n.
hortum 3
xortum ci fil[in] -u+.
şlang ci -la [ca(n)] (fışkırtmak, püskürtmek, sulamak).
burağan ci - [ca(d)] şiddetlenmek.
hücre 2
hüceyrä co 2
hücrä ci .
iç 2
iç ci c22[+accu] -.
çäk ci (sigara, tütün, nargile) -.
ikramiye 2
uduş ci (piyango, çekiliş)+(iles, abla) - (almak, kazanmak);(talihli, şanslı) - almak.
mükâfat ci ca(s) maaş -;şirket [3+dati] - vermek;(işçilere, memurlara, personele, çalışanlara) [ca(a)] - (almak, vermek).
iktidar 2
hökûmät co 25
iqtidar ci -(lı, sız) [ca(a)] 3;- yetmek.
ilahi 2
dini mahnı ci - (söylemek, çalmak, dinlemek, okumak);-yi.
ilâhi ci - [ca(a)] ca(n).
ilik 2
ilik co 2
ilgäk ci - (dikmek, açmak);-den geçirmek.
illet 2
illät co 2
zählätäkän ci - [ca(a)] ca(n).
işletme 2
istehsalı
planlaşdırma co 26
müässisä co 47
kalın 2

galın	ci - [ca(a)] ca(n).
başlıg	ci - vermek.
kalkan	2
galxana	
bänzär balıg	ci - balığı;- (avlamak, yakalamak, yemek, temizlemek,ayıklamak).
galxan	co 1
kalp	2
qälb	ci co3.
saxta	ci - para.
kanun	2
kanon	co 48
gânun	co 44
kap	2
gab	ci -a (dökmek, koymak, aktarmak, koymak);c23 -ı.
üz	ci ca(n)+geni -ı;-(ı, ını) örtmek.
kara	2
gara	ci - ca(n);-ca [ca(a)] ca(n).
guru yer	ci - parçası;-da yaşamak;-ya ayak basmak.
kavra	2
tut	ci 6f [ca(d)] -;-+dsav 6f.
gavra	ci 6t [ca(d)] -;-+dsav 6t.
kıy	2
qiymä-qiymä	
doğramak	ci c23+accu -.
gıy	ci ca(n)+dati -.
koca	2
är	ci o3+ps -+ps;-+dsnv.
böyük	ci - ca(n).
kol	3
gol	ci c51[+ps3s] -u;co3.
gol	ci -u (tutmak, çekmek, çevirmek).
şöbä	ci (daire, şirket)+geni [ca(a)] -u;-unda çalışmak.
kompleks	2
kompleks	ci - [ca(a)] c55;c55 [ca(a)] -.
ruhi nögsanlıg	co 2
kon	2
gon	ci c65 [ca(d)] -.
goyul	ci c66 [ca(d)] -.
kredi	2
borc	ci -(almak, vermek);- (miktarı, limiti, haddi, izni) artmak.
e'tibar	ci -+ps artmak;- kazanmak;3+geni [ca(a)] -.
kurşun	2
güllä	co 40
gurğuşun	co 13

kurum 2
gurum ci -lamak;- (bulaştırmak, olmak).
cämiyyät co 47
kuşak 2
gurşag ci -+ps bağlamak;co4.
näsil ci a6 - [ca(a)] c75.
kuyruk 2
guyrug ci 4[+ps3s] -u.
növbä ci -da beklemek;-a girmek;- olmak;c59+loca+pron [ca(a)]-.
lisans 2
âli tähsil co 43
diplom ci -[+ps][+accu] (almak, göstermek, var).
makam 2
melodiya ci c50 -ı.
mägam ci c76 -ı;-ında (karşılamak, ağırlamak, misafir etmek, kabul etmek).
mâni 2
bayatı co 30
mâne ci - olmak.
metin 2
mätn ci .
mätin ci - olmak.
muhtar 3
mähällä rähbärici co 2.
mähällä rähbäri ci - muhtarı.
muxtar ci - [ca(a)] (yönetim, devlet, idare).
nasılsa 2
haradansa ci - ca(v)+(pate, pite).
älbättä ci - ca(v)+(tlte, pite)+eech.
neden 2
nä üçün ci mood(int).
säbäb ci c(n)+geni -i;c(a) -.
nefis 2
näfs ci .
näfis ci - [ca(a)] ca(n).
not 2
qiymät co 43
qeyd ci - (tutmak, almak);
ocak 3
yanvar ci -ayı;ca(s) yılı -ı.
ocag co 5
ocag ci -da (pişirmek, ısıtmak).
olgun 2
kâmil ci co1.

yetişmiş ci - [ca(a)] c52;c52 [ca(d)] -[+dsav].
olumlu 2
tăsdiq ci - (fiil, cümle;fiil, cümle) -.
pozitiv ci - [ca(a)] c53;c53 [ca(d)] -[+dsav].
olumsuz 2
inkâr co 15
negativ ci - (fiil, cümle;fiil, cümle) -.
oyna 2
gımılda ci c22[yerinden] -.
oyna ci 3 [(parti, disco, düğün, nişan)]+loca [ca(d)] -;-yan 3.
oyun 2
oyun ci .
tamaşa co 56
pek 2
çox ci .
sărt ci - [bir] c67;c67 [ca(a)] -.
pirinç 2
bürünc co 13
düyü co 41
pişkin 2
utanmaz ci co1.
yaxşı bişmiş ci - [ca(a)] c46;c46 [ca(a)] -.
pul 2
pul ci c49[+ps3s] -u.
marka ci - (vermek, almak, saymak);
saf 2
saf ci .
săf ci .
sap 2
saplag ci 5[+ps] -si;5+loca+pron -.
sap ci c51[+ps3s] -ı;c51+loca+pron -.
satır 3
çapacag co 5
çapacag ci -la (kesmek, doğramak, dilimlemek).
sătir ci (alt, üst) -;-dan başlamak;-ı okumak;-a (yazmak,geçmek,çizmek, başlamak).
saz 2
saz ci -harf;harf -.
gamış ci co1.
sessiz 2
sâmit co 15
sâkit ci 3+ps -ı.
sıfat 2
sifăt co 15

keyfiyyät co 7
sıra 3
növba ci -da beklemek;-a girmek;- olmak;c59+loca+pron [ca(a)]-.
sıra ci -da oturmak;-dan kalmak.
sıra ci -ya sokmak;-lamak;-yla ca(v).
sıva 2
suva ci .
çirmä ci .
sinir 2
sinir co 2
äsäb ci 3+ps -[ler].
şekerleme 2
konfet ci - (yemek, hazırlamak, satmak).
müngülämä ci - yapmak.
şık 2
şıg ci - [ca(a)] 3.
alternativ ci -i (seçmek, tercih etmek, uygulamak).
taban 2
daban ci co3.
döşämä co 34
tabir 2
yazma ci -ci;- etmek;rüya -i.
tä'bir co 0
takım 3
dästä ci - [ca(a)] 3;3+plrp [ca(a)] -.
däst ci - [ca(a)] 2;2+plrp [ca(a)] -.
komanda ci c54 -ı.
tekne 2
gämi co 17
täknä ci hamur -si.
temsil 2
tamaşa co 56
tämsil ci - etmek.
tezgâh 2
däzgâh ci c57 -ı.
pištaxta co 58
tokmak 2
toxco 5
tokmak ci kapı -ı.
tulum 2
kombinezon co 16
tulug ci 4[+ps3s] -u.
tuş 2
toxmagcıg ci c60[+ps3s] -u.

güläşdä küräyi
yere däymä ci - (olmak, yapmak);-la (yenmek, kazanmak, yenilmek,kaybetmek, galip gelmek, mağlub olmak).
ünlü 2
sâit ci - harf;harf -.
şöhrätli ci - [ca(a)] 3.
vasat 2
orta ci - [ca(a)] ca(n);-(ı, ını) (al, ver);- olanı (al, ver).
şärâit ci ca(a) -.
vekâlet 2
nazirlik ci c69+dati - etmek;c69+geni -i.
väkâlät ci 3+dati - etmek;3+geni -i.
vur 2
güllälä ci c61+iles -.
vur ci c70+iles -;c70+accu [ca(d)] -;c77 -.
yaka 2
yaxa ci c78 -sı.
sâhil co 27
yalı 2
sâhildäki ev co 34
sâhil co 27
yapı 2
bina co 71
bünyä ci 3+ps -.
yargı 2
qärar ci -ya varmak;3 hakkındaki -;- belli olmak.
mähkämä işläri ci -ya intigal etmek;- (yürütmek, görülmek);- işi.
yazı 2
mäktub co 30
reşka ci paranın - (tarafı, yüzü).
yüz 2
yüz ci - [(tane, adet)] 7f -ü.
üz ci 3+ps;2+ps -ü;yüzüm;yüzün.
yüz 2
üz co 27
soy ci 4 (derisi[ni], postu[nu]) [ca(d)] -;derisi -+dsva 4.
zar 2
zar co 2
zär ci - [+accu] [ca(d)] atmak.

E.5. Macro Collocation Info Definitions Lexicon

Macro collocation info definitions lexicon is presented below in text form. Each line contains a single definition. The same notation is used in the collocation definitions as in Sections E.3 and E.4.

- [ca(a)] 3;3 [ca(a)] - [+dsav].
c75 -; - c75[+dsav];- (yemek,içmek,oynamak,konuşmak,koşmak).
1+ps -;- kanamak.
- (giymek,yıkamak,silmek).
- (avlamak,avlanmak,yemek,içmek).
- (uçmak,süzülmek,avlanmak,yakalanmak).
- (sürünmek,avlanmak,avlamak,vurulmak).
- (yazmak,okumak,eleştirmek,incelemek).
- içmek.
-+iles (avlamak,vurmak,öldürmek).
- (ekmek,yemek,büyümek).
- (çalmak,dinlemek).
- (oynamak,seyretmek,izlemek).
- yemek.
- (pişirmek,ısıtmak,hazırlamak).
benim -.
ca(a) -.
ca(d) -.
benim -+ps1s.
-an ca(n);ca(d) -an.
ca(d) -.
ca(a) -la;-la ca(v);-la (beraber,birlikte).
ca(d) -+pe2p;-+pe2p (dedik,diye);siz -+pe2p.
senin -+ps2s.
- ötmek.

E.6. Identical Suffixes Lexicon

The identical suffixes lexicon is presented below in text form. Each line contains the information for a single suffix or suffix sequence. In each entry the generic symbol for the suffix or the generic symbols for the suffixes in the suffix sequence, and its collocation info definition are stored. "ci" means a collocation definition follows. The same notation is used in the collocation definitions as in Sections E.3 and E.4.

```
accu                        ci p3 -;ca(n)+loca+pron -;
dsav1 dsva6                 ci - ca(n);ca(n) -[+dsav].
dsav1 dsvd1                 ci - ca(v).
dsna11                      ci - ca(n).
dsnd3                       ci - ca(v).
dsnv3                       ci ca(d) -.
dssa3                       ci - ca(n);ca(n) -.
dsva1                       ci - ca(n);ca(n) -.
dsvp3                       ci - ca(n);ca(n) -.
dsva7 dati                  ci ca(a) -;- doğru.
dsvd3                       ci (3+accu,3+dati) - ca(v).
dsvd6                       ci ca(n)+accu -;3 -.
dsvn3 ps2s                  ci senin -.
dsvn5 dsna3                 ci - ca(n);ca(n) -.
dsvn5 dsnn2                 ci co2.
dsvn5 ps1s                  ci benim [ca(a)] -.
dsvn5 ps2s                  ci senin -;ca(a) -.
dsvn5 ps2s dsna11           ci senin -.
dsvn5 ps2s dsnd3            ci senin -.
dsvn5 ps2s pe1p             ci senin -;biz -.
dsvn5 ps2p                  ci sizin -.
dsvn7                       ci p3 -.
dsvn7 dati                  ci p3 -.
dsvn11                      ci ca(a) -;- ca(a).
dsvn12                      ci - (olayı,durumu,hareketi).
dsvv2                       ci ca(n)+accu -.
dsvv5                       ci ca(n)+abla -.
dsvv6                       ci ca(d) -.
```

dsvv7		ci ca(n)+abla -.
dsvv8		ci ca(d) -.
dsvv9		ci ca(d) -.
dsvv11		ci ca(n)+abla -.
dsvv12		ci ca(d) -.
dsvv27		ci ca(a) -;- ca(a).
dsvv30		ci ca(d) -.
geni		ci p3 -;ca(n)+loca+pron -;- ca(n)+ps.
geni	pe1p	ci p3 -;ca(n)+loca+pron -.
negp		ci ca(d) -;ca(n)+accu -.
pe1s		ci ben -.
pe1p		ci biz -.
pe2p		ci siz -;ca(d) -;ca(n)+accu -.
ps1s		ci benim -.
ps2s		ci senin [ca(a)] -.
ps2s	case	ci senin [ca(a)] -.
ps2s	pe1p	ci senin [ca(a)] -.
ps2p		ci sizin -.
ps3s		ci onun [ca(a)] -.
ps3s	case	ci onun [ca(a)] -.
tlte		ci ca(n)(n) -;ca(d) -.

E.7. Representative Listing of the Bilingual Suffix Lexicon

A representative listing of the bilingual suffix lexicon is presented below. The lexicon contains the transitions for nouns and adjectives. The symbol # indicates that a word category follows, and the number 1/0 next to the word category indicates whether a word in this category can be a valid final word (1) or not (0). The suffixes that can be affixed to the words in this category are stored in the following lines. The information for each suffix is stored in two consecutive lines. The fields in the first line are in the following order:

1. Turkish suffix (an empty suffix corresponds to a free jump)
2. final state of the transition
3. flag associated with the suffix
4. flag to indicate whether the suffix is affixed to the stem separately (1) or not (0)
5. flag to indicate whether the suffix is a derivational suffix (y) or an inflectional one (c)
6. flag to indicate whether the last consonant of the suffix is affected by the consonant mutation rule (1) or not (0)
7. Generic suffix name
8. The equivalent Azeri suffix

In the next line the information for the vowel harmony rule and the consonant harmony rules are stored for the Turkish suffix and its equivalent Azeri suffix, respectively.

```
#n0 ,1
cık             ,n4 , 14,0 y 1 dsnn1        cık
110000 000000                               011000 000000
cı              ,a3 ,  6,0 y 0 dsna3        q
110000 000000
cı              ,n5 ,  6,0 y 0 dsnn2        çı
110000 000000                               010000 000000
et              ,v4 , 13,0 y 1 dsnv1        q
000000 000000
hane            ,n0 ,  1,0 y 0 dsnn4        xana
000000 000000                               000000 000000
kâr             ,a0 ,  5,0 y 0 dsna5        q
000000 000000
lık             ,a5 , 11,0 y 1 dsna6        lık
010000 000000                               011000 000000
lık             ,n2 , 11,0 y 1 dsnn7        lık
010000 000000                               011000 000000
name            ,n0 ,  2,0 y 0 dsnn8        nâmä
000000 000000                               000000 000000
sa              ,v5 , 12,0 y 0 dsnv2        q
010000 000000
                ,a0 ,  0,0
                ,n1 ,  0,0
#n1 ,1
lan             ,v9 , 17,0 y 0 dsnv3        q
010000 000000
laş             ,v4 , 18,0 y 0 dsnv4        q
010000 000000
la              ,v5 , 16,0 y 0 dsnv6        q
010000 000000
                ,n6 ,  0,0
#n2 ,1
cık             ,n1 , 14,0 y 1 dsnn10       cık
110000 000000                               011000 000000
cı              ,n3 ,  6,0 y 0 dsnn11       q
110000 000000
                ,n1 ,  0,0
#n3 ,1
lık             ,n1 ,  0,0 y 1 dsnn12       lık
010000 000000                               011000 000000
                ,n1 ,  0,0
```

```
#n4 ,1
cı              ,n5 ,  0,0 y 0 dsnn13        q
110000 000000
                ,n5 ,  0,0
#n5 ,1
lık             ,n1 ,  0,0 y 1 dsnn14        lık
010000 000000                                011000 000000
                ,n1 ,  0,0
#n6 ,1
lı              ,a0 ,  0,0 y 0 dsna9         q
010000 000000
sız             ,a0 ,  0,0 y 0 dsna10        q
010000 000000
                ,n7 ,  0,0
#n7 ,1
                ,n8 ,  0,0
                ,n9 ,  0,0
#n8 ,0
lar             ,n9 ,  0,0 c 0 plrp          q
010000 000000
#n9 ,1
ca              ,d1 , 21,0 y 0 dsnd1         q
110000 000000
ca              ,n22,  0,0 y 0 dsnd2         q
110000 000000
ımız            ,n10,  0,0 c 0 ps1p          q
101000 100000
ım              ,n10,  0,0 c 0 ps1s          q
100000 100000
ınız            ,n10,  0,0 c 0 ps2p          q
101000 100000
ın              ,n10,  0,0 c 0 ps2s          q
100000 100000
                ,n18,  0,0
                ,n22,  0,0
#n10,1
ca              ,a5 ,  0,0 y 0 dsna11        q
110000 000000
ca              ,d2 ,  0,0 y 0 dsnd3         q
110000 000000
sız             ,*  ,  0,0 y 0 dsnn15        q
010000 000000
                ,n18, 0,0
```

```
#n11,1
dan             ,n16,  0,0 c 0 abla         dan
110000 000000                               010000 000000
da              ,n12,  0,0 c 0 loca         da
110000 000000                               010000 000000
nın             ,n15,  0,0 c 0 geni         q
010000 100000
ya              ,n14,  0,0 c 0 dati         q
010000 100000
yı              ,n20,  0,0 c 0 accu         nı
010000 100000                               010000 100000
                ,n15,  0,0
#n12,1
ki              ,n13,  0,0 c 0 pron         q
000000 000000
                ,v43,  0,0
#n13,1
                ,n8 ,  0,0
                ,n16,  0,0
                ,n17,  0,0
#n14,1
da              ,n21,  0,0 c 0 loca         q
010000 000000
                ,v49,  0,0
#n15,1
ki              ,n13,  0,0 c 0 pron         q
000000 000000
                ,n16,  0,0
#n16,1
da              ,n21,  0,0 c 0 loca         q
010000 000000
                ,v44,  0,0
#n17,1
ndan            ,n16,  0,0 c 0 abla         q
001000 000000
nda             ,n12,  0,0 c 0 loca         q
001000 000000
nın             ,n15,  0,0 c 0 geni         q
010000 000000
na              ,n14,  0,0 c 0 dati         q
010000 000000
nı              ,n20,  0,0 c 0 accu         nı
010000 000000                               010000 100000
                ,n19,  0,0
```

```
#n18,1
                ,n11,  0,0
                ,n19,  0,0
#n19,1
yılan           ,d2 ,  0,0 c 0 iles          ?
010100 100000
yınan           ,d2 ,  0,0 c 0 iles          ?
010100 100000
yla             ,d2,   0,0 c 0 iles          q
001000 100000
#n20,1
da              ,n21,  0,0 c 0 loca          q
010000 000000
#n21,1
#n22,1
sı              ,n17,  0,0 c 0 ps3s          q
010000 100000
#a0 ,1
al              ,v8 , 38,0 y 0 dsav1         q
100000 100000
dan             ,d2 , 37,0 y 0 dsad1         dan
110000 000000                                010000 000000
ca              ,a3 ,  0,0 y 0 dsaa1         q
110000 000000
ca              ,d2 , 21,0 y 0 dsad2         q
110000 000000
en              ,d2 , 36,0 y 0 dsad3         ãn
000000 000000                                000000 000000
                ,a1 ,  0,0
#a1 ,1
                ,a2 ,  0,0
#a2 ,1
                ,a3 ,  0,0
#a3 ,1
lık             ,a4 , 11,0 y 1 dsaa3         lık
010000 000000                                011000 000000
                ,n0 ,  0,0
#a4 ,1
ına             ,d2 ,  0,0 y 0 dsad4         q
101000 100000
                ,n0 ,  0,0
#a5 ,1
                ,n6 ,  0,0
#a6 ,1
```

E.8. Representative Listing of the Turkish Proper Nouns Lexicon

A representative listing of the Turkish proper nouns lexicon, including its first 100 entries, is presented below in text form. Fields in an entry are separated by commas, and the information for each entry is given in a single line. The fields in each entry are in the following order:

1. Turkish proper noun
2. semantic features
3. flags

0 and 1 represent the false and true boolean values, respectively. Proper noun semantic features are in the following order:

human, nation, country, city(town, village, region), sea(ocean, lake, river), mountain, language

```
aba             ,1000000, 0111111100100
abaiang         ,0000000, 0000001110100
abak            ,1000000, 0111111100100
abakan          ,1000000, 0111111100100
abakay          ,1000000, 0111111100100
abamüslim       ,1000000, 0111111100101
abamüslüm       ,1000000, 0111111100101
abana           ,0000100, 0000001110100
abant           ,0000100, 0000001110100
abay            ,1000000, 0111111100100
abaza           ,0100000, 0111111100110
abaç            ,1000000, 0111111100100
abbas           ,1000000, 0111111100100
abbasi          ,0100000, 0000001101011
abd             ,0000000, 0000001110100
abdal           ,1000000, 0111111100100
abdi            ,1000000, 0111111100101
```

abdullah ,1000000, 0111111100100
abdülalim ,1000000, 0111111100101
abdülazim ,1000000, 0111111100101
abdülaziz ,1000000, 0111111100101
abdülbaki ,1000000, 0111111100101
abdülbari ,1000000, 0111111100101
abdülbasir ,1000000, 0111111100101
abdülbasit ,1000000, 0111111100101
abdülcabbar ,1000000, 0111111100101
abdülcebbar ,1000000, 0111111100101
abdülcelil ,1000000, 0111111100101
abdülcemal ,1000000, 0111111100101
abdülcevat ,1000000, 0111111100101
abdülezel ,1000000, 0111111100101
abdülferit ,1000000, 0111111100101
abdülfettah ,1000000, 0111111100101
abdülgaffar ,1000000, 0111111100101
abdülgafur ,1000000, 0111111100101
abdülgani ,1000000, 0111111100101
abdülhadi ,1000000, 0111111100101
abdülhak ,1000000, 0111111100101
abdülhakim ,1000000, 0111111100101
abdülhalik ,1000000, 0111111100101
abdülhalim ,1000000, 0111111100101
abdülhamit ,1000000, 0111111100101
abdülkadir ,1000000, 0111111100101
abdülkahhar ,1000000, 0111111100101
abdülkerim ,1000000, 0111111100101
abdüllatif ,1000000, 0111111100101
abdülmecit ,1000000, 0111111100101
abdülmelik ,1000000, 0111111100101
abdülmennan ,1000000, 0111111100101
abdülmesih ,1000000, 0111111100101
abdülmetin ,1000000, 0111111100101
abdülnasır ,1000000, 0111111100101
abdülvahap ,1000000, 0111111100101
abdülvahit ,1000000, 0111111100101
abdürrahim ,1000000, 0111111100101
abdürrahman ,1000000, 0111111100101
abdürrauf ,1000000, 0111111100101
abdürrezzak ,1000000, 0111111100101
abdürreşit ,1000000, 0111111100101
abdüssamet ,1000000, 0111111100101
abdüssami ,1000000, 0111111100101

abdüsselam	,1000000,	0111111100101
abdüssemih	,1000000,	0111111100101
abdüssettar	,1000000,	0111111100101
abdüzzeki	,1000000,	0111111100101
abha	,0000000,	0000001110100
abid	,1000000,	0111111100101
abide	,1000000,	0111111100111
abidin	,1000000,	0111111100101
abidjan	,0000000,	0000001110101
abil	,1000000,	0111111100101
abit	,1000000,	0111111100101
ablak	,1000000,	0111111100110
abraş	,1000000,	0111111100100
abuzer	,1000000,	0111111100101
abuzettin	,1000000,	0111111100101
abuşka	,1000000,	0111111100100
abır	,1000000,	0111111100100
aca	,1000000,	0111111100100
acabay	,1000000,	0111111100100
acabey	,1000000,	0111111100101
acahan	,1000000,	0111111100100
acapulco	,0001000,	0000001110100
acar	,1000000,	0111111100100
acaralp	,1000000,	0111111100100
acarbey	,1000000,	0111111100101
acarer	,1000000,	0111111100101
acarkan	,1000000,	0111111100100
acarlar	,0100000,	0000001110100
acarman	,1000000,	0111111100100
acarsoy	,1000000,	0111111100100
acartürk	,1000000,	0111111100100
acaröz	,1000000,	0111111100101
acatay	,1000000,	0111111100100
accra	,0000000,	0000001110100
acem	,0100000,	0111111100111
aclan	,1000000,	0111111100100
acun	,1000000,	0111111100110
acunal	,1000000,	0111111100100
acunalan	,1000000,	0111111100100

APPENDIX F. SAMPLE RUNS FOR THE TRANSLATOR

Two types of sample runs for the translator are given below. First examples of single sentence translations are presented, and then an example of text translation is given. For each Turkish sentence the time it takes for the morphological parser to parse each word in the sentence (mtime) are presented together with the total translation time. All the results are in seconds.

Turkish sentence : Ali eve geldi.
Azeri sentence : Ali evã gãldi .

mtime = 0.39
mtime = 0.17
mtime = 0.11

total time for translation = 2.10

Turkish sentence : ayağını yıkadı.
Azeri sentence : ayağını yudu .

mtime = 0.66
mtime = 0.22

total time for translation = 2.31

Turkish sentence : uydu yörüngede dönüyor.
Azeri sentence : peyk orbitada dönür .

mtime = 0.11
mtime = 0.16
mtime = 0.11

total time for translation = 2.42

Turkish sentence : bu anahtar kapıya uydu.
Azeri sentence : bu açar gapıya uydu .

mtime = 0.17
mtime = 0.44
mtime = 0.44
mtime = 0.17

total time for translation = 4.73

Turkish sentence : komutan alayı teftiş etti.
Azeri sentence : komandir alayı täftiş etdi .

mtime = 0.27
mtime = 0.44
mtime = 0.28
mtime = 0.17

total time for translation = 6.43

Turkish sentence : komutan askerle alay etti.
Azeri sentence : komandir äsgärlä lağ etdi .

mtime = 0.27
mtime = 0.22
mtime = 0.27
mtime = 0.22

total time for translation = 6.20

Turkish sentence : alaydan hiç hoşlanmaz.
Azeri sentence : (alay / lağ) heç xoşlanmaz .

mtime = 0.49
mtime = 0.6
mtime = 0.33

total time for translation = 4.27

Turkish sentence : pastan battım.
Azeri sentence : pasdan batdım .

mtime = 0.38
mtime = 0.17

total time for translation = 1.93

Turkish sentence : senin pastan batmış.
Azeri sentence : sänin tordun batmış.

mtime = 0.27
mtime = 0.44
mtime = 0.11

total time for translation = 5.10

Turkish sentence : yetmiş tane elbise sattık.
Azeri sentence : yetmiş dänä paltar satdıg .

mtime = 0.22
mtime = 0.27
mtime = 0.22
mtime = 0.22

total time for translation = 4.83

Turkish sentence : bu senin arabanın kapısı.
Azeri sentence : bu sänin arabanın gapısı .

mtime = 0.6
mtime = 0.33
mtime = 0.38
mtime = 0.44

total time for translation = 5.87

Turkish sentence : Hasan elmasını yedi.
Azeri sentence : Hasan almasını yedi .

mtime = 0.38
mtime = 0.77
mtime = 0.16

total time for translation = 3.35

An example of text translation is presented below. Translation time for each sentence in the text are given together with the overall translation time of the whole text. All the results are in seconds.

Turkish text:

sabah erkenden kalktım. kahvaltıdan sonra evden çıktım. arabamla işe vardım. oldukça yorucu bir gündü. akşam saatleri yollar sıkışıktı. dolayısıyla eve geç gelebildim. böylece gün bitmiş oldu.

Azeri text:

sähär erkändän galxdım. qähväaltıdan sonra evdän çıxdım. arabamla işä vardım. oldugca yorucu bir gündü. axşam saatları yollar sıxışıkdı. dolayısıyla evä gec gäläbildim. beläcä gün bitmiş oldu.

translation time for the 1st sentence = 2.90
translation time for the 2nd sentence = 3.73
translation time for the 3rd sentence = 5.06
translation time for the 4th sentence = 6.21
translation time for the 5th sentence = 7.30
translation time for the 6th sentence = 3.19
translation time for the 7th sentence = 3.57

overall translation time for the text = 31.98
avreage translation time for a sentence = 4.57

REFERENCES

1. Hutchins, W. J., *Machine Translation: Past, Present, Future, Ellis Horwood*, 1986

2. Nirenburg, S., J. Carbonell, M. Tomita and K. Goodman, *Machine Translation: A Knowledge-Based Approach*, Morgan Kaufmann Publishers, California, 1992.

3. Özgüven, M. K. and J. Tsujii, "An Approach to Machine Translation," *Proceedings of the First Turkish Symposium on Artificial Intelligence and Artificial Neural Networks*, Ankara, 1992.

4. Stoop, A. M., "Transit in the World of Machine Translation: Towards an Automatic Translator for Dutch and Turkish," *Proceedings of the Third Conference on Turkish Linguistics*, Tilburg, 1987.

5. Stoop, A. M., "ATMACA: Semantic Analysis by the Computer," *Proceedings of the Fourth Conference on Turkish Linguistics*, Ankara, 1990.

6. Hankamer, J., "Morphological Parsing and the Lexicon," *Lexical Representations and Processing*, ed. W.M. Wilson, MIT Press, 1988.

7. Kibaroğlu, M. Okan, "Spelling Checking in Agglutinative Languages and an Implementation for Turkish," M.S. Thesis, Boğaziçi University, 1991.

8. Akın, H. L., S. Kuru, T. Güngör, İ. Hamzaoğlu and D. Arbatlı, "A Spelling Checker and Corrector for Turkish," *Proceedings of the Second Turkish Symposium on Artificial Intelligence and Artificial Neural Networks*, İstanbul, 1992.

9. Hankamer, J., "Finite State Morphology and Left to Right Phonology," *Proceedings of the West Coast Conference on Formal Linguistics*, Vol. 5, Stanford University, 1986.

10. Solak A. and K. Oflazer, "Parsing Agglutinative Word Structures and Its Application to Spelling Checking for Turkish," *Proceedings of the Fifteenth International Conference on Computational Linguistics*, Vol. 1, pp. 39-45, Nantes, 1992.

11. Özgüven, M. K., personal communication.

12. Koç, N., *Yeni Dilbilgisi (The New Grammar)*, İnkılap Yayınları, İstanbul, 1990 (in Turkish).

13. Ercilasun, A. B. and A. M. Aliyev, *Karşılaştırmalı Türk Lehçeleri Sözlüğü (Comparative Dictionary of Turkish Dialects)*, Kültür Bakanlığı, Ankara, 1991 (in Turkish).

14. Bozkurt, F., *Türklerin Dili (The Language of Turks)*, Cem Yayınları, İstanbul, 1992 (in Turkish).

15. Güngör, T. and S. Kuru, "Representation of Turkish Morphology in ATN," *Proceedings of the Second Turkish Symposium on Artificial Intelligence and Artificial Neural Networks*, İstanbul, 1992.

16. Oflazer, K. "Two-Level Description of Turkish Morphlogy," *Proceedings of the Second Turkish Symposium on Artificial Intelligence and Artificial Neural Networks*, İstanbul, 1992.

17. Güngördü, Z. and K. Oflazer, "A Lexical Functional Grammar for a Subset of Turkish," *Proceedings of the Second Turkish Symposium on Artificial Intelligence and Artificial Neural Networks*, İstanbul, 1992.

18. Tın, E. and V. Akman, "Resolution of Pronominal Anaphora in Turkish," *Proceedings of the Second Turkish Symposium on Artificial Intelligence and Artificial Neural Networks*, İstanbul, 1992.

19. Boguraev, B. and T. Briscoe, *Computational Lexicography for Natural Language Processing*, Longman, 1990.

20. Sowa, J. F., "Logical Structures in the Lexicon," *Knowledge-Based Systems*, Vol. 5, No. 3, 1992.

21. Tomabechi, H., "Direct Memory Access Translation," *Proceedings of the 10th International Joint Conference on AI(IJCAI87)*, pp. 722-725, Milano, 1987.

22. Nogier, J. F. and M. Zock, "Lexical Choice as Pattern Matching," *Knowledge-Based Systems*, Vol. 5, No. 3, 1992.

23. Charniak, E. and D. McDermott, *Introduction to Artificial Intelligence*, Addison-Wesley, 1985.

24. Kuru, S. and H. L. Akın, Internal Design Specification, Spelling Checking Software for Turkish to be Integrated with Dec ALL-In-1 Office Automation Package, Department of Computer Engineering, Boğaziçi University, 1992.

25. Pamuk, O., *Kara Kitap (Black Book)*, Can Yayınları, İstanbul, 1990 (in Turkish)

REFERENCES NOT CITED

Gazdar G. and C. Mellish, *Natural Language Processing in Prolog*, Addison-Wesley, 1989.

Guenthner, F., H. Lehmann and W. Schönfeld, "A Theory for the Representation of Knowledge," *IBM Journal of Research and Development,* Vol. 30, No. 1, 1986.

Jacobs, P. S., "TRUMP : A Transportable Language Understanding Program," *International Journal of Intelligent Systems*, Vol. 7, pp. 245-276, 1992.

Guo, C., "Interactive Vocabulary Acquisition in XTRA," *Proceedings of the 10th International Joint Conference on AI(IJCAI87)*, pp. 715-717, Milano, 1987.

Bookman, L. A., "A Microfeature Based Scheme for Modelling Semantics," *Proceedings of the 10th International Joint Conference on AI(IJCAI87)*, pp. 715-717, Milano, 1987.

Sezer, A., "Turkish Adjective Clauses on Computer," *Studies on Turkish Linguistics*, pp. 565-577, Middle East Technical University, Ankara, 1988.

Koç, S., "Turkish Adverb Clauses on Computer," *Studies on Turkish Linguistics*, pp. 579-596, Middle East Technical University, Ankara, 1988.

Türkçe Sözlük (Turkish Dictionary), Cilt I, Türk Dil Kurumu, Ankara 1988 (in Turkish).

Türkçe Sözlük (Turkish Dictionary), Cilt II, Türk Dil Kurumu, Ankara 1988 (in Turkish).