EXAMPLE BASED MACHINE TRANSLATION FROM ENGLISH TO TURKISH WITH SYNCHRONOUS STRUCTURED STRING-TREE CORRESPONDENCE (SSTC)

A MASTER'S THESIS

in

Computer Engineering

Atılım University

By
N.Deniz ÖZTÜRK
August 2007

EXAMPLE BASED MACHINE TRANSLATION FROM ENGLISH TO TURKISH WITH SYNCHRONOUS STRUCTURED STRING-TREE CORRESPONDENCE (SSTC)

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE

OF

ATILIM UNIVERSITY

BY

N.Deniz ÖZTÜRK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF COMPUTER ENGINEERING

August 2007

Approval of the Graduate School of Natural of Applied Sciences

Prof.Dr.Selçuk SOYUPAK

Director

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

Prof.Dr.İbrahim AKMAN

Head of Department

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

Asst.Prof.Dr.Çiğdem TURHAN

Supervisor

Examining Committee Members

Asst.Prof.Dr.Nergiz Ercil ÇAĞILTAY	
Asst.Prof.Dr.Murat KOYUNCU	
Instructor Fügen SELBES	
Asst.Prof.Dr.Çiğdem TURHAN	
Hüseyin EROL	

ABSTRACT

EXAMPLE BASED MACHINE TRANSLATION FROM ENGLISH TO TURKISH WITH SYNCHRONOUS STRUCTURED STRING-TREE CORRESPONDENCE (SSTC)

N.Deniz ÖZTÜRK
M.S. in Computer Engineering
Supervisor: Assist. Prof. Dr. Çiğdem TURHAN
August, 2007

Example based machine translation is a translation method that uses previous translations. Its system requires information built up with pre-translation of source language and its corresponding equivalent in target language. With a new sentence entry to system, method searches the translation part of corpus as a whole or as parts with matching technique and collects the corresponding results in target language. Then, these parts are aligned accordingly to grammatical rules of target language. In this work uses tree search algorithm and methods for matching and alignment steps. This structure provides great flexibility in many subjects.

Keywords: Example Based Machine Translation, Machine Learning, Synchronous Structure String Tree (SSTC).

ÖZET

İNGİLİZCEDEN TÜRKÇEYE SENKRONİZE YAPISAL SIRA AGAÇ YAPISI İLE ÖRNEK TABANLI OTOMATİK ÇEVİRİ

N.Deniz ÖZTÜRK
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yard. Doç. Dr. Çiğdem TURHAN
Ağustos, 2007

Örnek tabanlı otomatik çeviri sistemleri daha önce yapılmış olan çevirileri kullanan bir bir sistemdir. Sistem çeviri yapmak için kaynak dil ile ona karşılık gelen hedef dil cümlelerinden oluşan bilgi birikiminden faydalanılır. Sistem çeviri yapacağı yeni cümleyi hedef dilin bulunduğu veritabanında bütün ya da parçalar halinde arayarak uygun bölümlerini ve ona karşılık gelen hedef dildeki sonuçlarını hafızaya alır. Daha sonra karşılık gelen parçaları uygun şekilde birleştirip hedef dil yapısına göre düzenler. Bu çalışmada veritatabanı oluşturmak için yapısal ağaç yapısını kullanmaktadır. Bu yapı birçok konuda esneklik sağlamaktadır.

Anahtar Sözcükler : Örnek Tabanlı Otomatik Çeviri, Otomatik Öğrenme, Senkronize Yapısal Sıra Agaç Yapısı.

To my baby

ACKNOWLEDGMENTS

I express sincere appreciation to my supervisor Asist. Prof. Dr. Çiğdem TURHAN for her guidance and insight throughout the research.

I would like to thank my family, especially my mother Dudu ÖZTÜRK and my little brother Serhat ÖZTÜRK for supporting and believing in me throughout my life.

I would like to thank to my husband Ahmet ALP for his great support, encouragement and understanding while writing this thesis.

I would like to thank Özgür SOĞUKPINAR for her help and great support.

In addition, I would like to thank to my colleagues at Turkish Land Forces Army and Tepeteknolojik Servisler Inc.for their moral support while writing this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Machine translation (MT) is a sub-field of computational linguistics that tries to translate text or speech from one natural language to another with computer software. MT attempts to automate all, or part of the process of translating from one human language to another [1].

When it was first introduced, MT was an unthinkable goal; but today a number of systems exist which produce output meaningful translations in a number of specific domains. In the MT sector, “European Association for Machine Translation (EAMT)" is the organization that serves people that are interested in MT and translation tools [2].

1.1 Why MT Matters

MT is an important topic, socially, politically, commercially, scientifically, and philosophically and its importance is likely to increase in the future [1]. The social or political importance of MT appears most in communities where more than one language is generally spoken, for example the European Union. Translation is necessary for communication and gathering information. There are few human translators, and there is a limit on the amount of translation that can be performed without automation.

Following factors give the commercial importance of MT. First of all, translation itself is commercially important because, when someone is choosing a product he must select between products including instruction manual written in his mother language, or manual written in English. If the first case is true, the chance of using the new product effectively will be higher. Additionally, translation is expensive. Translation is a high skilled job, requiring much more than knowledge of the given languages. Moreover, loss of time during translation is costly. Finally, world wide usage of the Internet forces us to translate a document on the web between different languages in a time and cost effective manner.

Scientifically, MT is an application area which is a testing ground for many ideas in Computer Science, Artificial Intelligence, and Linguistics. Some of the most important developments in these fields have begun in MT. Philosophically; MT shows an attempt to automate an activity that requires human knowledge. Improved output quality can also be achieved by human intervention: for example, some systems can translate more accurately if the user has unambiguously identified which words in the text are names. MT has proven to be useful as a tool to assist human translators and in some cases can even produce output that can be used "as is".

1.2 Thesis Outline

In this thesis, the Example Based Machine Translation (EBMT) method is used to produce translations from English to Turkish language. The bilingual corpus is used on MS Office help documents. Thesis proposes a structure tree to store the corpora. In the next chapter, a summary about MT is given and in Chapter 3, the EBMT system is explained. Chapter 4, gives the details of the SSTC and in Chapter 5 the design and implementation information for the ORHUN system architecture along with the evaluation results of the performance of the system is provided. The thesis ends with Conclusion and Future Work in Chapter 6.

CHAPTER II

MACHINE TRANSLATION

Machine Translation (MT) is a branch of computational linguistics that researches on the use of computer software to translate text or speech from one natural language to another. Simple MT systems generate a translation by performing basic substitution of words in source language to target language. More complex translations can be attempted by sign corpus techniques; these techniques give better results for the differences in linguistic typology, phrase recognition, and translation of idioms [3].

There are many reasons to use MT. The first reason is the speed of translation as MT tools translate much quicker than humans. In addition, computers can make similar translation continuously; however, human translators require variety; for example: technical materials may become too boring for human translators. Sometimes human translation is not needed at all, as required quality in translation may be low. In such situations MT would be best solution as computers do not produce good translations. Plus, translation cost can be lowered with MT.

2.1 History of MT

The research on MT started when Warren Weaver wrote a paper on using computers for translation in July 1949 [4]. Later a group of scientist held a conference on MT in 1952. The first translation system was shown in Jan.1954 [4].

A few years later a system was installed by IBM at the US Air Force in 1959, and in 1963 and 1964, Georgetown University, installed systems at Euratom and at the US Atomic Energy Agency. In the mid 1950's Russia started its MT research [4].

The Systran system was started at the US Air Force in 1970, which translated Russian to English. The Météo system for translating weather forecasts was started in Canada in 1976 [5], which was later bought by The European Commission Systran system. Japanese companies began the production of commercial systems in the 1980's and the first translator workstations were put up for sale in 1990. Nowadays, MT has become an online service on the Internet [5].

Some examples of MT products are [6]:

1 Systran (http://babelfish.altavista.com/)
2 Comprendium (based on Metal)
3 ProMT (http://www.translate.ru/eng)
4 ESTeam (memory-based)
5 http://ourworld.compuserve.com/homepages/WJHutchins/Compendium-4.pdf
6 http://www.foreignword.com/Technology/mt/mt.htm

2.2 MT System Architectures and Strategies

In this section, MT strategies will be explained. MT has three main approaches: Direct Translation, Rule Based Translation, and Corpus Based Translation. In addition, Corpus Based Translation has two strategies: Statistical and EBMT [7]. Figure 2.1 shows the strategies used in MT:

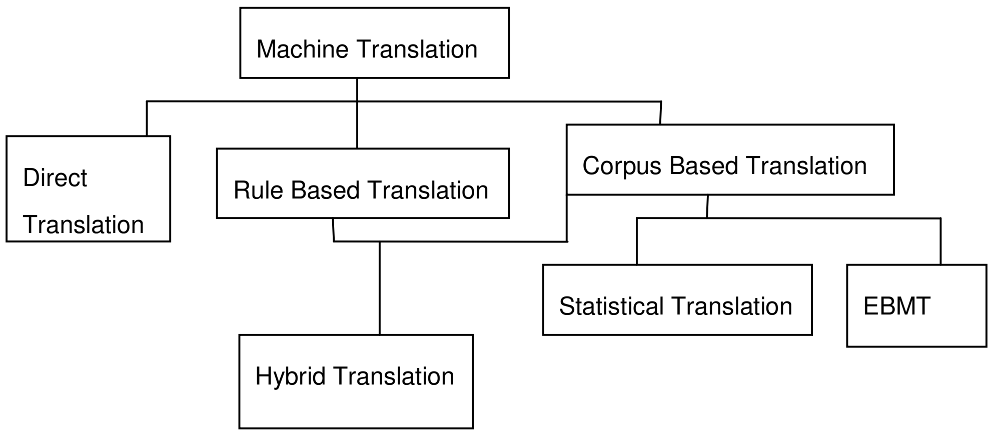


Figure 2.1. Machine Translation Strategies

2.2.1 Direct Translation

A direct translation system is designed for a particular source and target language. This system does not need to analyze the sentence structurally or morphologically. The system translates word-by-word so the translation system uses large dictionaries. Some simple grammatical adjustments e.g. on word order and morphology are later performed on the words translated. Systran is basically a direct translation system.

2.2.2 Rule-Based Translation

This approach requires the analysis and representation of the 'meaning' of source language and the generation of the equivalent target language. Rule-Based MT has two major approaches: The first one transfer and approach operates in three stages - analysis into source language, transfer to target language, and generation or synthesis of the target language.

The second approach is the Interlingua (IL) which has three steps. The first step translates source texts into abstract language independent description; the second step translates this description into equivalent target language; and finally generates the target language text. The IL system supposes that it is possible to translate source texts to more than one language with abstract representation. Translation is thus in two stages: from the source language to the IL and from the IL to the target language. Esperanto language is used in some interlingua systems [12].

In addition, another IL language example used is Eurolang which is used in the translation between ten language pairs such as English-French-German, Italian and Spanish, and French-German[12].

2.2.3 Corpus-Based MT

This method supposes that the reference translations from source language to target language have equivalent meanings. These systems presume with given sample texts, one can achieve suitable source language/target language linking mechanism. This system can only be generated if suitable example translation set is available in the sample text. Corpus Based system includes two methodologies: statistical and example based MT.

Corpus-Based approaches are differentiated with direct use of information. This information is derived from corpora for the analysis, transfer and generation of translations and indirect use of corpora that is used as source of information for deriving or compiling lexical, grammatical and knowledge databases, and as sources of statistical information about source and target languages.

Examples of methodologies that employ direct use of information are statistics-based, example-based and connectionist approaches. Lexical and knowledge acquisition and the use of statistical information in other rule-based systems are examples of indirect use of corpora.

2.2.4 Statistics-Based MT

The basis of the method is to use of the probabilities that any one word in a sentence of one language corresponds to two, one or zero words in the translated sentence in the other language. The IBM research on this type is based on a large corpus of the Canadian Hansard, which records parliamentary debates in both English and French[12].

2.2.5 Example-Based MT

The basic argument for Example Based MT (EBMT) is that, translation involves finding or recalling analogous examples, discovering or remembering how a particular source language expression or something similar has been translated before. A more detailed analysis of EBMT can be found in the next section.

2.2.6 Hybrid Methods

Hybrid methods merge the rule-based approaches and the corpus-based methods.

CHAPTER III

EXAMPLE BASED MACHINE TRANSLATION

EBMT is an acronym for Example-Based Machine Translation. EBMT is based on the use of translation of a sentence that has been previously translated in the corpora. Turkish sentence is in the form of SOV (Subject, Object, Verb) like Japanese grammar structure. EBMT has used by Japans scientist successfully. So EBMT is selected this work.

EBMT must use a bilingual corpus which consists of the source and target languages and samples dependent on the domain. EBMT consists of three phases: matching, alignment, and recombination [10]. EBMT starts with matching using a matching algorithm that finds the examples that are most similar as a whole or part to the input sentence. Then the alignment step rebuilds the input sentence using a combination algorithm through the combination of found is performed. Lastly, recombination transfer and composition algorithm extracts corresponding target parts and combines them into a sentence in the target language.

Figure 3.1 shows EMBT architecture. As shown in the figure, a sentence is taken from source language and is searched in the BKB (Bilingual Knowledge Bank) for a sentence or part of sentence that is similar. If a similar sentence is found, corresponding part of the sentence is taken from the target text in the BKB. Lastly, the recombination procedure is performed to achieve the translation into the target sentence.

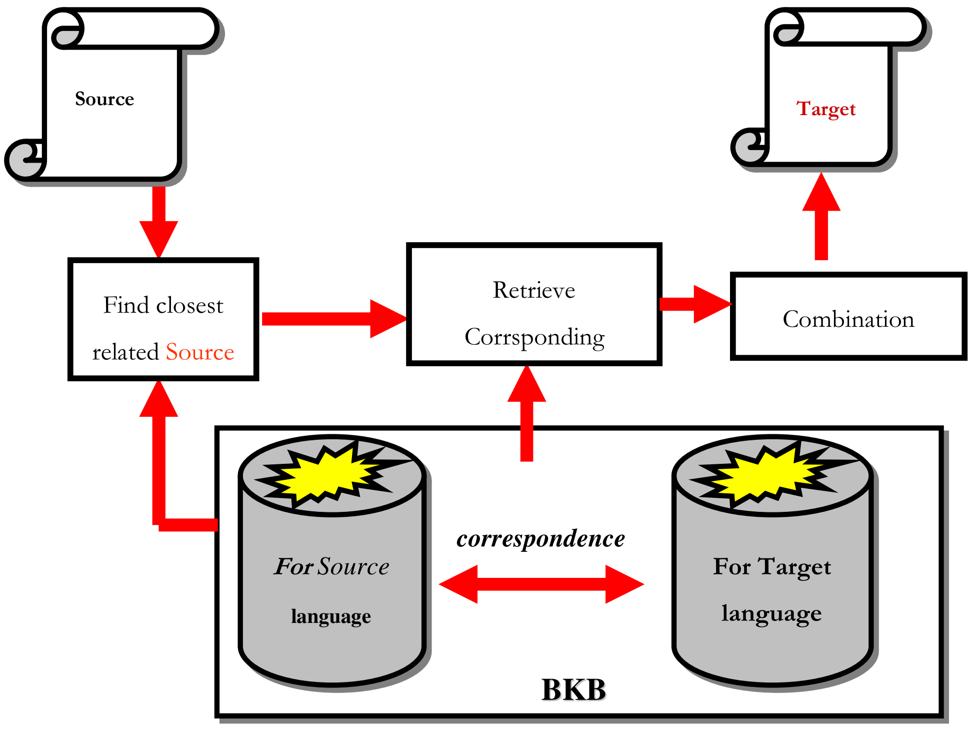


Figure 3.1 EBMT architecture

3.1 History of EBMT

EBMT has been initially mentioned in the paper by Makoto Nagao in 1981 and in a conference in 1984 [12]. EBMT is called "machine translation by example-guided inference or machine translation by the analogy principle as described by Nagao [12].

"Man does not translate a simple sentence by doing deep linguistic analysis, rather, man does translation, first, by properly decomposing an input sentence into certain fragmental phrases .., then by translating these phrases into other language phrases, and finally by properly composing these fragmental translations into one long sentence. The translation of each fragmental phrase will be done by the analogy translation principle with proper examples as it references" [12].

3.2 EBMT Methodology

EBMT has three main parts: matching phase to find a suitable example from database of real examples of source language, a translation phase, and then recombining these to give the target text. This is explained by a pyramid diagram as seen in Figure 3.2, which is first used by Vauquois in 1968 (pyramid has been adapted for EBMT [12]). In conventional MT make analysis whereas in EBMT perform matching. The input is the same example or examples set. First phase is *matching* where the suitable example or examples have been selected. The matching process is controlled by the semantic similarity between the lexical items in the input text and the corresponding items in the database.

In the next phase, if the system finds a suitable example, it selects corresponding parts in the target text. This phase is named *alignment* or *adaptation* and it is the same as the transfer method in conventional MT. If suitable parts have been selected, they must be *combined* in a way to achieve a target text, which is like the generation phase of conventional MT.

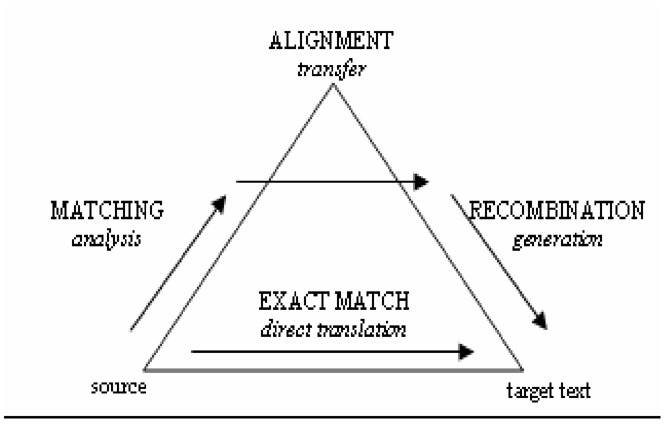


Figure 3.2. The “Vauquois pyramid” adapted for EBMT

EBMT has a lot of different extensions such as Memory-Based Translation (MBT), Transfer-Driven Machine Translation (TDMT) and Case-Based Machine Translation (CBMT).

1) She likes chocolate.	
Matches	
2)	a) She likes ice cream.
	O dondurma sever.
	b) I eat chocolate.
	Ben çikolata yerim.
Result	
3)	She likes chocolate.
	O çikolata sever

Figure 3.3 Example of EBMT

The example above Figure 3.3 shows that resulting in the translation of (1) can be a combination of the appropriate parts from (2a,b) so the system takes (3). First of all the system must take suitable parts of the example sentence.

3.2.1 Parallel Corpora

EBMT uses parallel corpora to compare source language and target language text. The following are examples of some parallel corpora; "The Canadian and Hong Kong parliaments have bilingual corpora in the form of their parliamentary proceedings, the European Union has multilingual documents and World Wide Web pages are available in two or more languages". After suitable corpus is built, the problem of alignment must be solved, i.e. identifying which parts (typically sentences) corresponds to each other [12].

3.2.2 Granularity of Examples

Length and similarity problems appear in the first step of EBMT matching step. These problems are: the longer the matched texts, the lower the probability of a complete match or the shorter the texts, the greater the probability of ambiguity and the greater the danger that the resulting translation will be of low quality, due to text boundary friction and incorrect chunking [12].

3.2.3 Number of Examples

There are reported issues to be addressed when parallel corpora is established such as the number of examples. One experiment, constructed by Mima et al. (1998) [12] showed how the quality of translation improved as more examples were added to the database. To test cases of the Japanese adnominal particle construction, they loaded the database with 774 examples in increments of 100. Translation accuracy increased steadily from about 30% with 100 examples to about 65% with the complete set. A similar result was found with another trial, rising from about 75% with 100 examples to nearly 100% with all 689 examples. Sumita & Lida (1991) [51] and Sato (1993) [52] also suggest that adding examples improves performance. This experiment reported that the improvement was more or less linear; it showed that there is some limit after which further examples cannot better the quality. After some limit, if the size of examples are increased the performance starts to decrease.

3.2.4 Suitability of Examples

A lot of EBMT systems work from a manually constructed database of examples, or from a carefully filtered set of “real” examples. Because, a huge corpus of naturally occurring text will contain overlapping examples of two sorts: some examples will reinforce each other by shows the identical translation phenomenon, however other examples will be in conflict. Like Somers et al., 1994 [31]; Öz & Cicekli, 1998[38]; Murata et al., 1999 [53] have mentioned, systems comprise a similarity metric which is sensitive to frequency, so that a large number of similar examples will increase the score given to certain matches.

3.2.5 Storage of Examples

EBMT systems have different storage types. Clearly, the storage types are related to the problem of searching for matches. One of these storage types is where the examples are stored as pairs of strings with no additional information. If example database is very large, indexing techniques are required, so Information Retrieval (IR) must be used.

3.2.5.1 Annotated Tree Structures

First storage method, which stores the examples as fully, annotated tree structures with explanation and links. Figure 3.4 shows how the Japanese sentence (3.1) "*She has long hair*" and its English translation are represented [14].

3.1) Kanojo wa kami ga nagai.
 she topic hair subj is-long
 'She has long hair.'

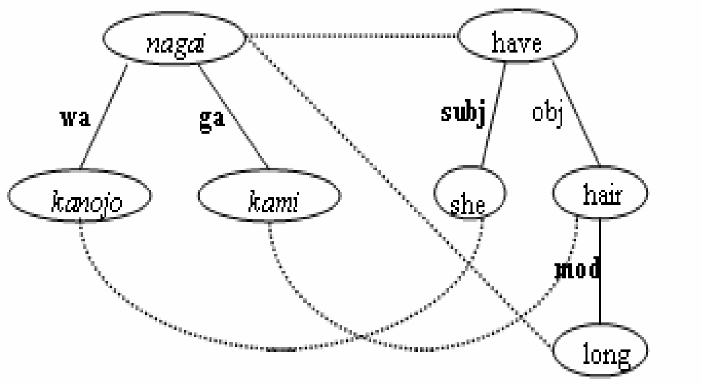


Figure 3.4. Representation scheme for (3.1)

More recently, a similar approach has been used by Poutsma (1998) and Way (1999): there, the source text is parsed using Bod's (1992) DOP (data-oriented parsing) technique, which is itself a form of example-based approach, then matching subtrees are combined in a compositional manner. In the system of Al-Adhaileh & Kong (1999) [15], examples are stored in dependency structures with links at the structural and lexical level expressed by indexes. Figure 3.5 shows the representation for the English–Malay pair in (3.2).

3.2 a. He picks the ball up.
b. Dia kutip bola itu

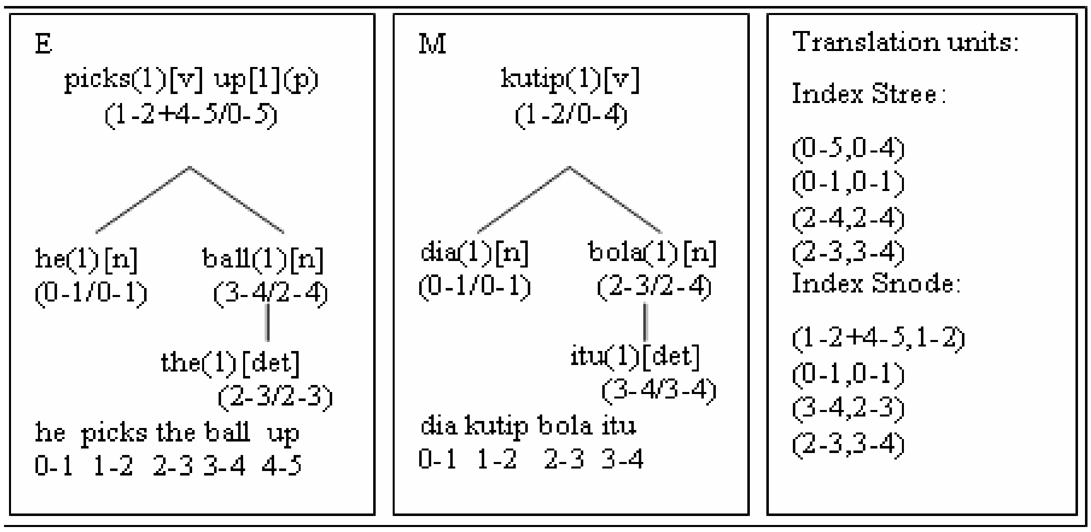


Figure 3.5. Representation scheme for (3.2)

The nodes in the trees are indexed to represent the lexical head and the span of the tree of which that item is the head: So for example the node labeled "ball(1)[n](3-4/2-4)" indicates that the subtree is headed by ball, which is the word spanning nodes 3 to 4 (i.e. the fourth word) when is the head of the subtree spanning nodes 2 to 4, i.e. the ball. The box labeled "*Translation units*" gives the links between the two trees, divided into "Stree" links, which identifies subtree correspondences (e.g. the English subtree 2-4 the ball corresponds to the Malay subtree 2-4 bola itu) and "Snode" links, which identifies lexical correspondences (e.g. English word 3-4 ball corresponds to Malay word 2-3 bola).

3.2.5.2 Generalized Examples

Some systems combined similar examples and store them as a single “generalized” example. Brown [16] tokenizes the examples to show equivalence classes such as “person’s name”, “date”, “city name”, and also linguistic information such as gender and number. In this method, sentence’s part in the examples is changed by these tokens, so the examples are more general. For example, (3.3 a) can be generalized as (3.3 b), and even, further as (3.3 c). If the system has an input like (3.3 d), this can be matched quite easily with (3.3. c) which can then be used as a template, whereas a match with the original text (3.3a) would be more difficult because of the differences.

3.2) a. John Miller flew to Frankfurt on December 3rd.
b. <1stname><lastname>flew to<city>on<month> <ord>.
c. <person-m> flew to <city> on <date>.
d. Dr Howard Johnson flew to Ithaca on 7 April 1997.

3.2.5.3 Statistical Approaches

For this storage technique, the basic structure which occurs more statistically in a language is stored in the database, so that the probability of translating a source text structure into a target text structure can be computed.

3.3 Matching

EBMT system starts to translate by finding the suitable example in the source language database. This search problem depends on how the examples are stored. For example; in the statistical method, the problem is essentially mathematical: maximizing a huge number of statistical probabilities.

3.3.1 Character-Based Matching

Matching techniques are needed in distance or similarity calculation. Firstly, the measure may be character-based pattern-matching in the examples that are stored as strings. In the earliest MT systems, only exact matches were possible [17], for example (3.4 a) would be matched with (3.4 b), but the match in (3.5) would not be found because the system has no way of knowing that small and large are similar.

3.4) a. This is shown as A in the diagram.
b. This is shown as B in the diagram.

3.5) a. The large paper tray holds up to 400 sheets of A3 paper.
b. The small paper tray holds up to 300 sheets of A4 paper.

3.3.2 Word-Based Matching

This measure was suggested by Nagao [18] and was used in some EBMT systems. This measure is based on similarity of meaning or usage. In the example, this measure can match if words in the input string are replaced by near synonyms. Nagao's measure is particularly effective in choosing between competing examples, as in Nagao's examples, where, given 3.6 a,b as models, the system choose the correct translation of "eat" in 3.7 as a taberu "eat (food)", in 3.6 b as okasu "erode", based on the relative distance from he to man and acid, and from potatoes to vegetables and metal.

3.6) a. A man <u>eats</u> vegetables. *Hito wa yasai o taberu.*
b. Acid eats metal. *San wa kinzoku o okasu.*

3.7) a. He <u>eats</u> potatoes. *Kare wa jagaimo o taberu.*
b. Sulphuric acid eats iron. *Ryūsan wa tetsu o okasu.*

3.3.3 Carroll's "Angle of Similarity"

This measure suggested by Carroll [19] is a trigonometric similarity measure based on both the relative length and relative contents of the strings to be matched. This measure searches for similar words and takes account of deletions, insertions and substitutions in the database for a given example. It evaluates a cost and the cost can be programmed to build linguistic generalizations.

3.3.4 Structure-Based Matching

This method requires examples that are stored as structured objects like a tree, and it involves tree-matching [20]-[23]. The "tree edit distance" can be used for tree comparison. Utsuro [24] attempted to reduce the computational cost of matching by taking the advantage of surface structure of Japanese language, in particular its case-frame-like structure. They developed a similarity measure based on the head nouns. Their method relies on the verbs matching exactly, and is limited to Japanese or similarly structured languages.

3.3.5 Partial Matching For Coverage

In the techniques mentioned above, it has been assumed that the aim of the matching process is to find a single example or a set of individual examples that provides the best match for the input. Another method is found in Nirenburg et al [25]. The matching function decomposes the cases, and makes a collection of respective terminology – "substrings", "fragments" or "chunks" of matched material.

3.4 Alignment (Adaptability) and Recombination

After matching, the system must translate the separated fragments (alignment or adaptation), and combine these parts to build a grammatical target output (recombination). This step has two problems: one of them is decide which part of the associated translation corresponds to the matched parts of the source text, and the other is recombining these parts in a suitable manner.

If the examples are stored as tree structures with the explicit correspondences between the parts, the alignment problem automatically disappears. For example, in Sato [26], the recombination stage is a kind of tree unification, Watanabe [27] on the other hand, adapts a process called "gluing" from Graph Grammars, which is a form of graph unification. Al-Adhaileh & Tang [15] state that the process is "analogous to top-down parsing".

Even if the examples are not annotated with the relevant information, in many systems the underlying linguistic knowledge includes information about correspondence at word or chunk level. This may be because the system makes use of a bilingual dictionary [29] or existing MT lexicon, as in the cases where EBMT has been incorporated into an existing rule-based architecture, e.g. Sumita [30]. Alternatively, some systems extract automatically from the example corpus information about probable word alignments, [31]-[34].

3.4.1 Boundary Friction

Boundary friction is a problem that some languages like Japanese and English have little or no grammatical inflection to indicate syntactic function [34]. For example, the translation associated with "*the handsome boy*" retrieved, from 3.8 is equally reusable in either of the sentences in 3.9. This however is not the case for a language like German (and like Turkish), where the structure of the determiner, adjective and noun can all carry inflections to indicate grammatical case, as in the translations of 3.9 a,b, shown in 3.10.

3.8 The handsome boy entered the room.

3.9 a. <u>The handsome boy</u> ate his breakfast.

b. I saw the handsome boy.

3.10 a. Der schöne Junge ass seinen Frühstück.

b. Ich sah den schönen Jungen.

A hybrid system can solve this problem. This system has a grammar of the target language, so it can take the results of the gluing process and somehow smooth them over.

3.4.2 Adaptability

Can all examples be equally reusable with their notion of “adaptability”? This question is asked by Collins & Cunningham [35]. Their example-retrieval method includes a measure of adaptability which shows that the similarity of the example has internal structure and external context. The notion of “adaptation-guided retrieval” has been developed in Case-Based Reasoning (CBR) [36]. In this notation, cases are retrieved from the example-base. It is not only their similarity with the given example, but also the extent to which they show a good model for the desired output. i.e. to which they can be adapted, that determines whether they are chosen. EBMT system, ReVerb, stores the examples together with a functional annotation, cross-linked to indicate both lexical and functional equivalence. Therefore, example-retrieval has two steps: the closeness of the match between the input text and the example, and the adaptability of the example, on the basis of the relationship between the example and its translation.

3.4.3 Statistical Modeling

Another approach to recombination is the statistical system where recombination is depicted as a statistical modeling problem. There is the “language model” that is invoked, with which the system tries to maximize the product of the word-sequence probabilities. This approach suggests that “recombined” target-language proposals could be verified: the frequency of co-occurrence of sequences of 2, 3 or more words (ngrams) can be extracted from database. If the target-language corpus is bigger, then appropriate statistics about the probable "correctness" of the proposed translation could be achieved. There are well-known techniques for calculating the probability of n-gram sequences, and a similar idea is found in Grefenstette's [37] experiment, mentioned above, in which alternative translations of ambiguous noun compounds are verified by using them as search terms on the World Wide Web.

3.5 Computational Problems

Above mentioned methods have to be implemented as computer programs, and significant computational factors influence many of them. One criticism to be made of the methods which store the examples as complex annotated structures is storage and matching/retrieval algorithms. This is especially problematic if such resources are difficult to obtain for one (or both) of the languages, as Güvenir & Cicekli [38] report, relating to earlier work by Güvenir & Tunç [39] on Turkish.

Other criticism is that the complexities involved decrease from some of the alleged advantages of EBMT, especially the idea that the system's linguistic knowledge can be extended "simply" by increasing the size of the example-set [40]: adding more examples becomes a significant overhead and then these examples must be parsed, and the results should be possibly checked by a human. Another advantage of the EBMT method is that system can be developed despite the lack of resources such as parsers, lexicons and so on.

One important computational issue is speed, particularly for real-time speech translation [41]. Massively Parallel Processors was used a small example base (1,000 cases) and achieved processing speeds almost 13 times faster than a more conventional architecture. For a more significant database, say 64,000 examples, the improvement would be 832 times. They warn however that speed advantages can be lost if the communication between the parallel processors and other processors is inefficient. Researchers want to find ways of maximizing the effect of the examples by identifying and making explicit significant generalizations. Hybrid systems solve the problem by combining the advantages of both the example-based and rule-based approaches.

CHAPTER IV

EXAMPLE-BASED MACHINE TRANSLATION BASED ON THE SYNCHRONOUS SSTC

In this chapter, a method for EBMT will be explained. This method creates a link between the string in a language and its representation tree structure which can be interpreted for both analysis and generation in MT. In Figure 4.1 a flexible annotation structure called Structured String-Tree Correspondence (SSTC) is given which was declared in Boitet & Zaharin [42] to record the string of terms, its combined representation structure and the corresponding sentence, which is stressed by the sub-sentence corresponding part of a SSTC.

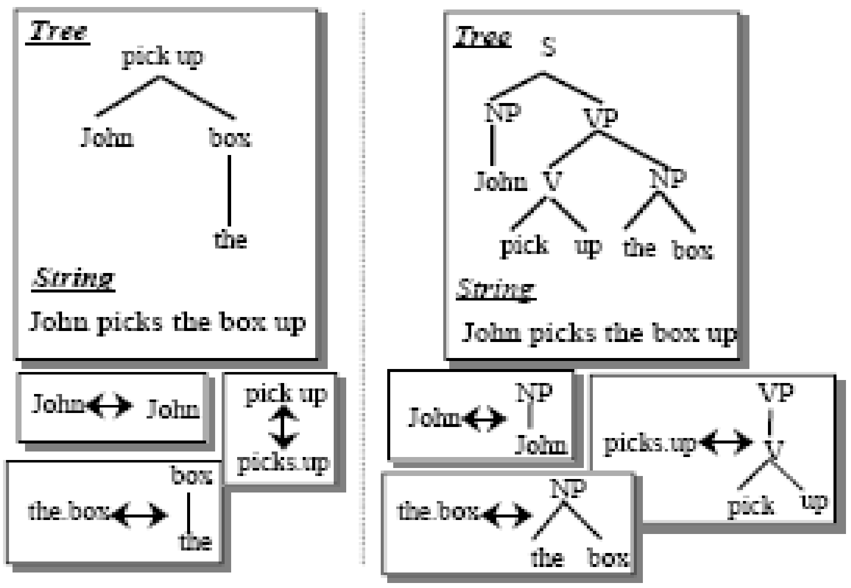


Figure 4.1 The correspondence between the string "*he picks the box up*" and its representation tree

Figure 4.1 is showed that the correspondence between the string "*he picks the box up*" and its representation tree (dependency tree and phrase structure tree), together with the sub-correspondences between the substrings and subtrees

As mentioned above the SSTC is a general structure that can combine a tree structure to string in a language which is the interpretation structure of the string, and more importantly is the facility to specify the correspondence between the string and the associated tree which can be nonprojective [42]. These features are very important in the design of an annotation scheme, in particular for the treatment of linguistic phenomena e.g. crossed dependencies [43]

SSTC can be summarized as a triple (*st, tr, co*), where **st** is a sentence or part of the sentence in source or target language, *tr* is its combined representation tree structure and *co* is the correspondence between st and *tr. co* between a string and its representation tree is made of two type correspondences between nodes in a tree and substrings in a string or between subtrees and substrings. The correspondence (co) is shown in each node N in the representation tree as two sequences of INTERVALS called SNODE(N) and STREE(N). SNODE(N) is an interval of the sub sentence in the sentence that corresponds to the node N in the tree. STREE(N) is an interval of the sub sentence in the sentence that corresponds to the subtree having the node N as root [44].

The notation used in SSTC demonstrates a correspondence consisting of a pair of intervals X/Y linked to each node in the tree, where X(SNODE) demonstrates the interval containing the substring that corresponds to the node, and Y(STREE) demonstrates the interval containing the substring that corresponds to the subtree having the node as root. This statement shows which word/s of the text corresponds/s to which node in the tree. For describing a NL using SSTC, a set of constraints were defined to govern such correspondences [28].

X:SNODE and *Y:STREE* intervals are governed by the following constraints:
i) *Global correspondence***:** an entire tree corresponds to an entire sentence.
ii) *Inclusion***:** a subtree which is part of another subtree T, must correspond to a substring in the substring corresponding to T.
iii) *Membership***:** a node in a subtree T, must correspond to a word which is member of the substring corresponding to T [44].

Figure 4.2 shows the sentence **"***The young girl looked***"** with its corresponding SSTC. It is a simple projective correspondence. An interval is assigned to each word in the sentence, i.e. (0-1) for "*The*", (1-2) for "*young*", (2-3) for "*girl*" and, (3-4) for "*looked*"

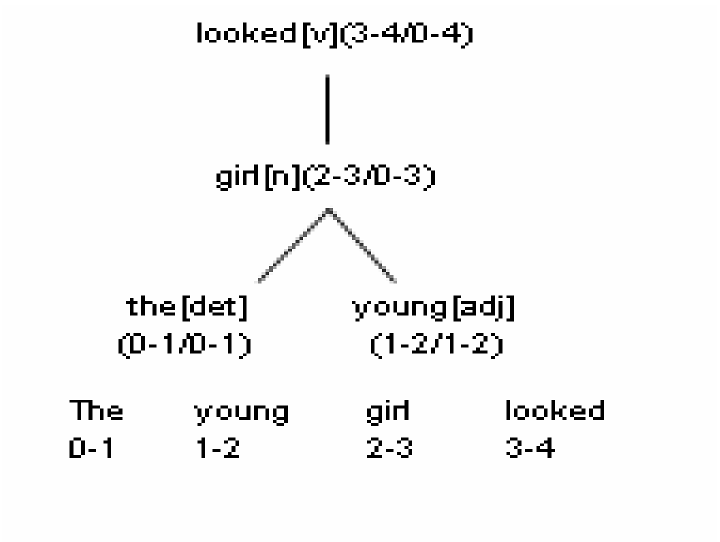


Figure 4.2 Representing a "The young girl looked" sentence and SSTC tree

A substring in the sentence which corresponds to a node in the representation tree is indicated by assigning the interval of the substring to SNODE of the node, e.g. the node "*girl*" with SNODE interval (1-2) corresponds to the word "*girl*" in the string with the like interval. The correspondence between subtrees and substrings are indicated by the interval assigned to the STREE of each node, e.g. the subtree rooted at node "eat" with STREE interval (0-4) corresponds to the whole sentence "*the young girl looked*".

4.1 Constructing a Bilingual Knowledge Bank Based on the Synchronous SSTC

Bilingual Knowledge Bank (BKB) containing bilingual parallel texts which links between the source and the target sentences is popular in implementing such EBMT systems. Sentences in the BKB are explained with their constituency or dependency structures, which supply the correspondences assembled at the structural level. The system chooses the dependency structure as the linguistic representation of the SSTC as it gives a natural way to set up the translation units between the *source* (English) and *target* (Turkish) SSTCs. In addition, it is believed that in order to increase the performance of the MT system, the SSTC structure can be extended to keep multiple levels of linguistic information. With this approach, translation examples are established by means of a synchronous SSTC editor as illustrated in Figure 4.3 [15].

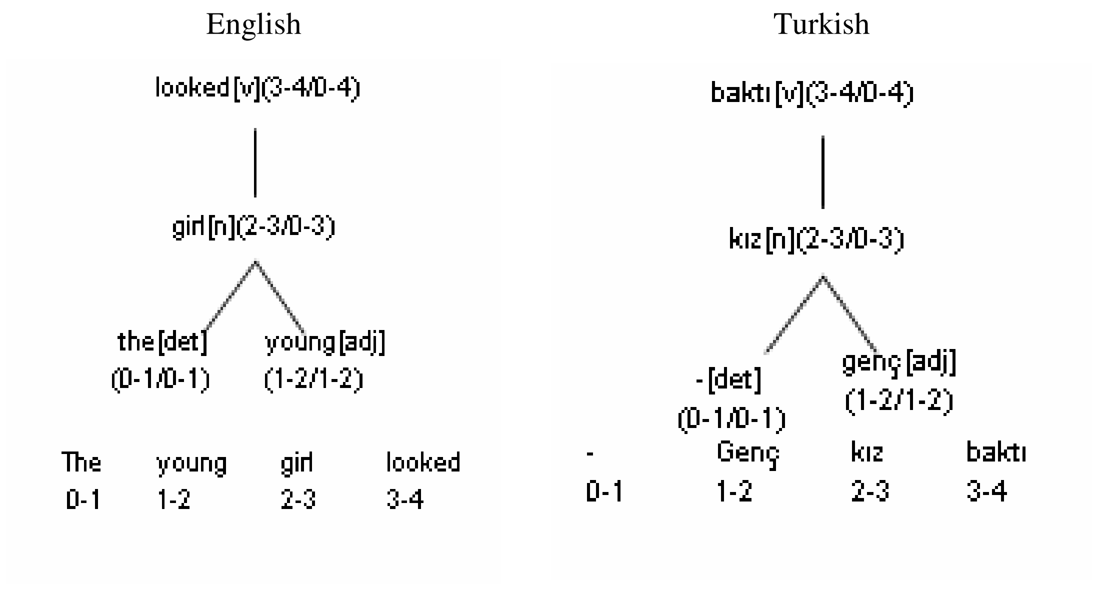
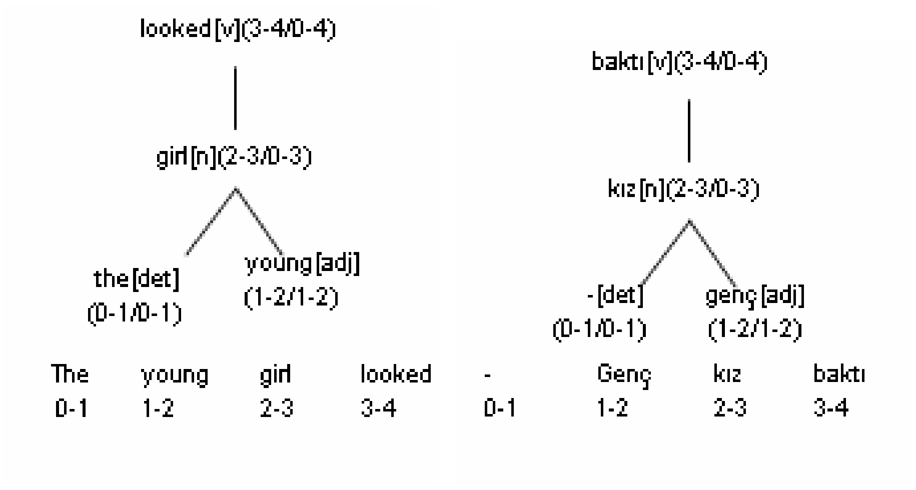


Figure 4.3 Represent “The young girl looked” and its SSTC and Turkish translation sentence “Genç kız baktı” and SSTC

The translation units are indicated in terms of STREE pairs (for phrases) and SNODE pairs between the *source* (English) and the *target* (Turkish) SSTCs. For example, as shown by the synchronous SSTC given in Figure 4.3, the fact that "*looked*" is translated to "*baktı*" is expressed by (3-4, 0-4) under the index SNODE of the translation units. Whereas, the fact that "*the young girl*" is translated to "*genç kız*" is expressed by (0-3, 0-3) under the index STREE of the translation units. In Figure 4.4, index of the SNODE and STREE can be clearly understood.

English Turkish Index



Index Stree
(0-4,0-4)
(0-1,0-1)
(1-2,1-2)
IndexSnode
(3-4,3-4)
(2-3,2-3)
(0-1,0-1)

Figure 4.4 Index of English and Turkish Sentence

The main characteristic of SSTC is flexibility. That flexibility is required in establishing translation units between source and target substrings which can be possibly discontinuous in both cases.

4.2 Example-Based Machine Translation Based On the SSTC

Figure 4.5 shows the process of translation of a *source* sentence into a *target* sentence [15]. The tagged *source* sentence must be parsed to establish a single rooted representation tree, based on the example-based parser. The parser builds the sub-SSTCs for all phrases in the *source* sentence by referring to some closely related examples in the BKB. To each sub-SSTC built for the *source* sentence, the corresponding *target* sentence sub-SSTC can be designated based on the translation units as established by the synchronous SSTCs in the BKB.

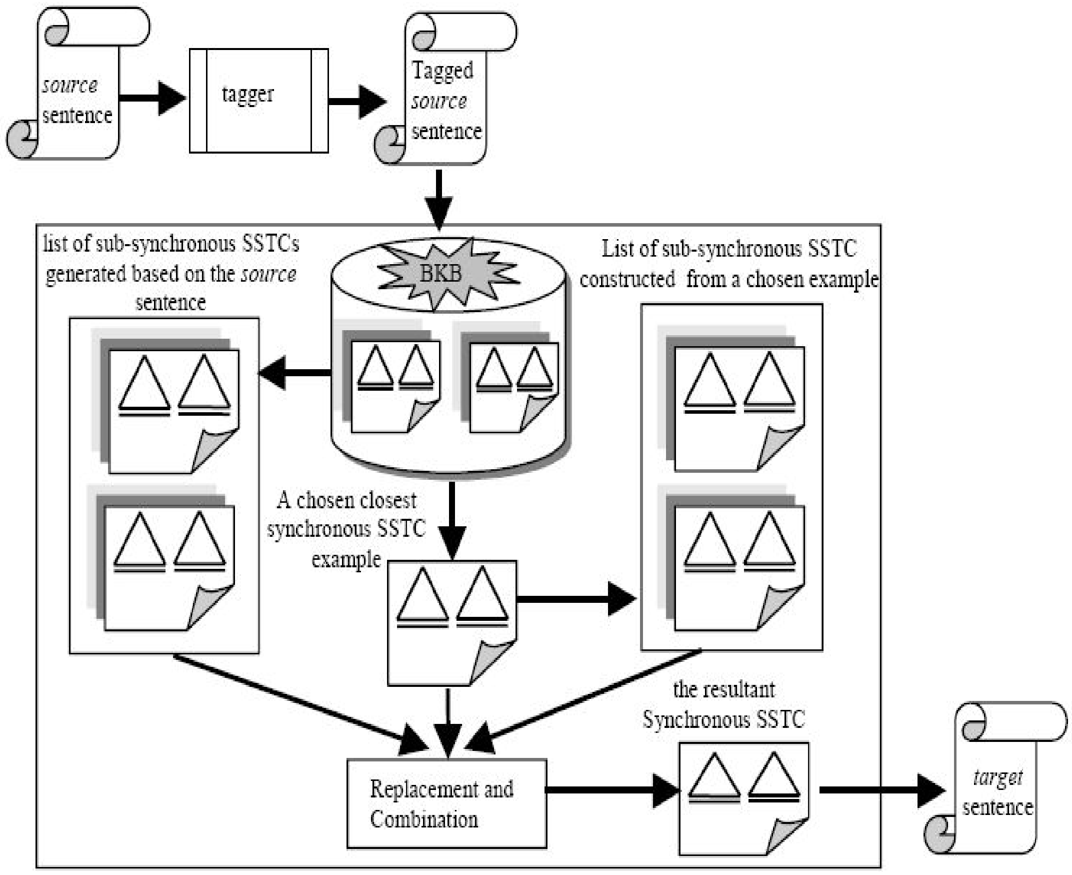

source sentence
tagger
Tagged source sentence
list of sub-synchronous SSTCs generated based on the source sentence
BKB
List of sub-synchronous SSTC constructed from a chosen example
A chosen closest synchronous SSTC example
the resultant Synchronous SSTC
Replacement and Combination
target sentence

Figure 4.5 Example-Based Machine Translation Based on the Synchronous SSTC

The *source* and *target* sub-SSTCs constituted together with the corresponding translation units identified will be formed as a list of sub-synchronous SSTCs. A list of sub synchronous SSTCs is constructed from the selected example, which will be replaced by their corresponding sub-synchronous SSTCs constituted from the *source* sentence to form the final synchronous SSTC through a combination process. Lastly, the *target* sentence as shown in the *target* SSTC will be obtained as an output as the translation of the *source* sentence. In the following part, an example to illustrate the process of translation will be presented as described above. Suppose the system intends to find the translation for the English *source* sentence *"the young girl bought a green dress"*, based on the set of examples represented in the example-base (BKB) in Figure 4.6.

First of all the system finds examples in the BKB which includes a word in the *source* sentence appearing as the root word of the dependency tree in its *source* SSTC. If more than one example are found (in most cases), the system must calculate the distance between the *source* sentence and these examples. The closest example (namely the one with minimum distance) will be chosen as a reference to combine the list of generated sub-synchronous SSTCs to form a complete synchronous SSTC [15].

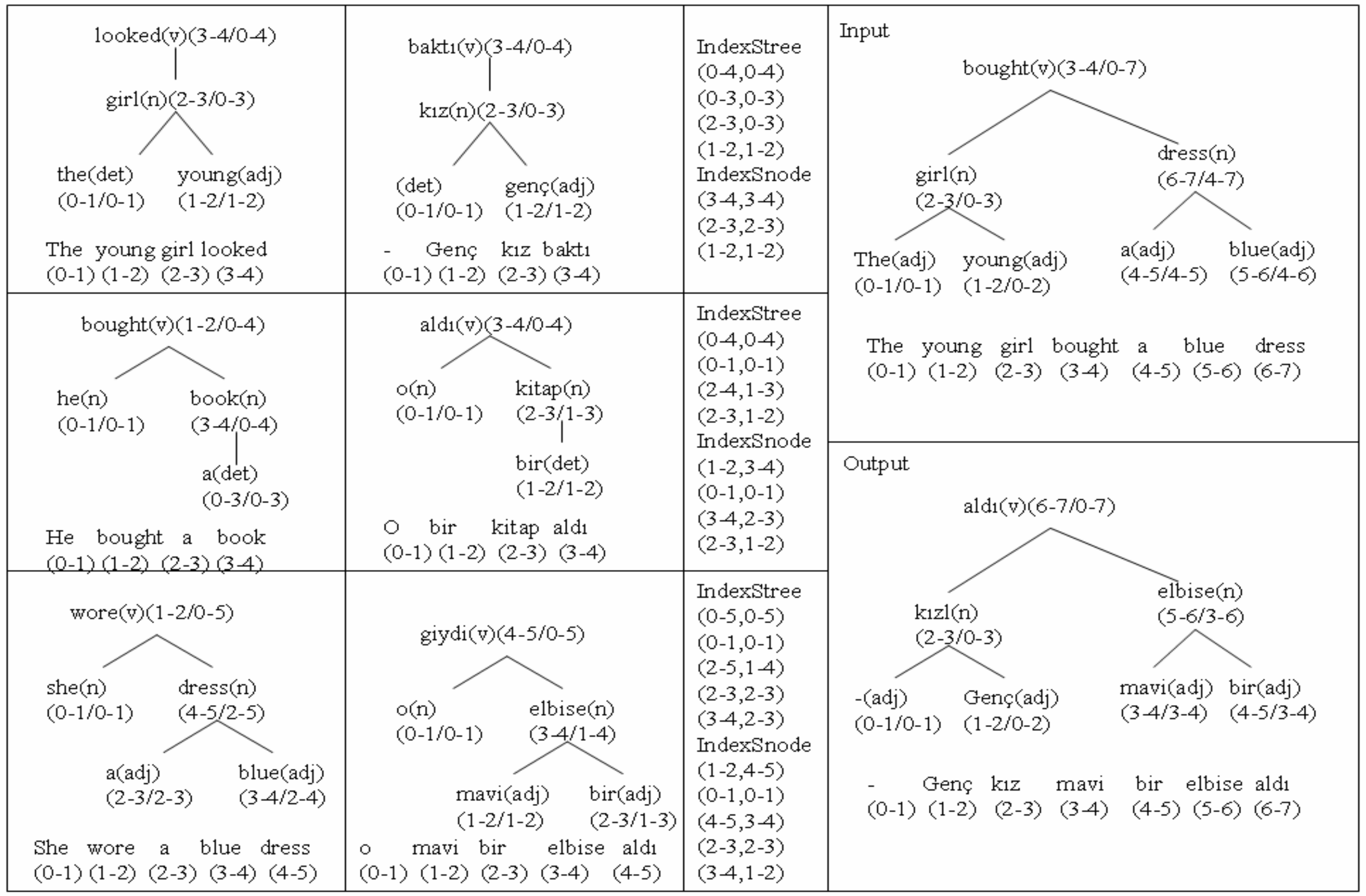


Figure 4.6 A set of synchronous SSTCs containing the English sentences, the Turkish translation sentences and their translation units

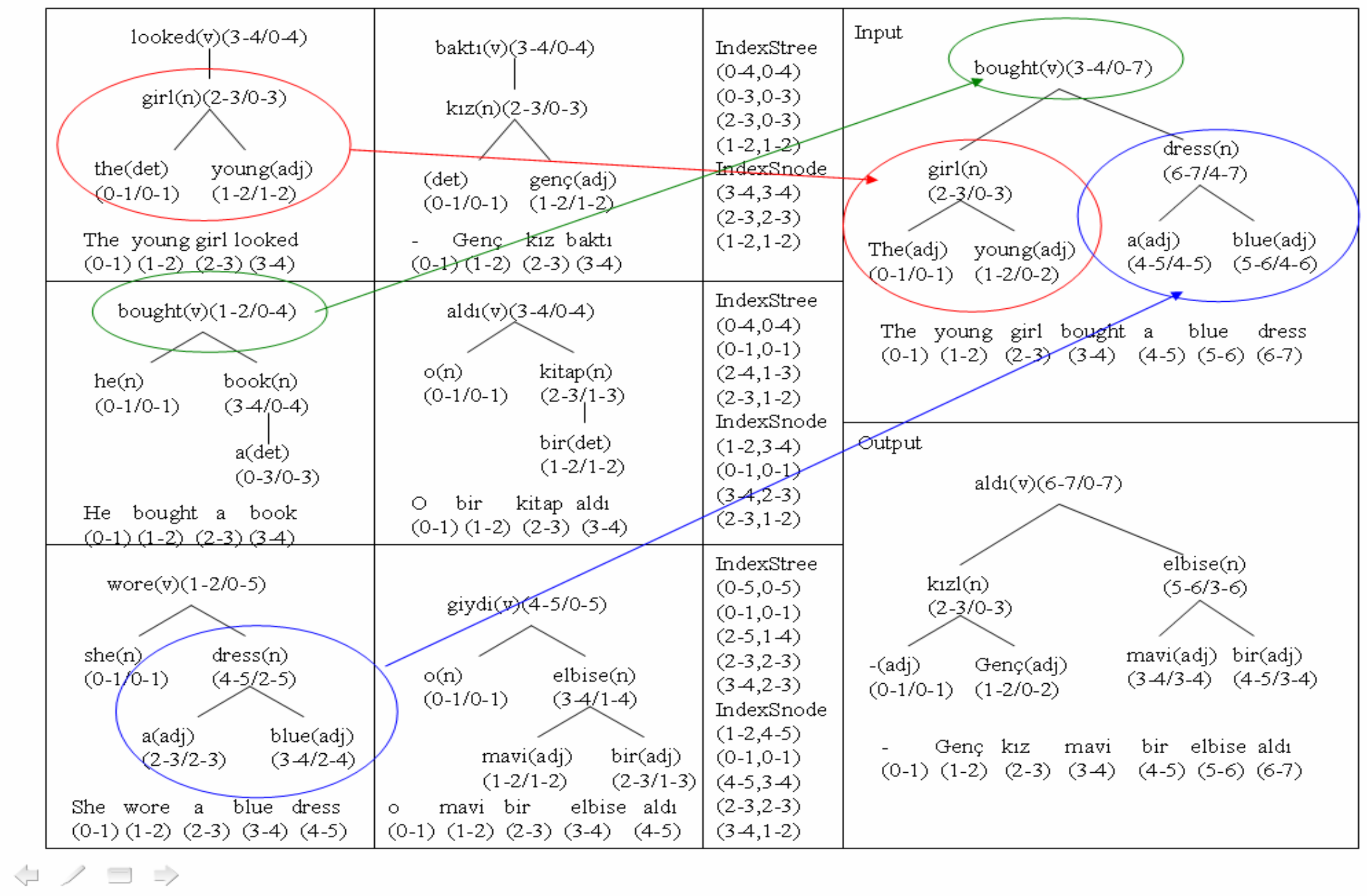


Figure 4.7*:* The replacement on the synchronous SSTC example and the translation for the *source* sentence "the ***young girl bought a blue dress***".

The sub-synchronous SSTCs computed from example (E, T) will be replaced by the corresponding sub-synchronous SSTCs built from the *source* sentence to form a complete synchronous SSTC. This approach is like the top down parsing technique. Consequently, the *target* sentence revealed in the *target* SSTC has been produced as the translation of the *source* sentence, i.e. "*The young girl bought a blue dress*" as illustrated in Figure 4.7

In the next step, the system must re-organize the target SSTC, since the English sentence structure is different than the Turkish sentence structure. Basically, simple English sentence grammar is SVO (Subject, Object, Verb), however, Turkish sentence is in the form of SOV (Subject, Object, Verb). If the system does not align, the translation sentence, "*Genç kız aldı bir mavi elbise*" will be produced. After the alignment process, based on the simple Turkish grammar rule, the system can produce "Genç kız bir mavi elbise aldı". Figure 4.8 shows the re-organized target SSTC result.

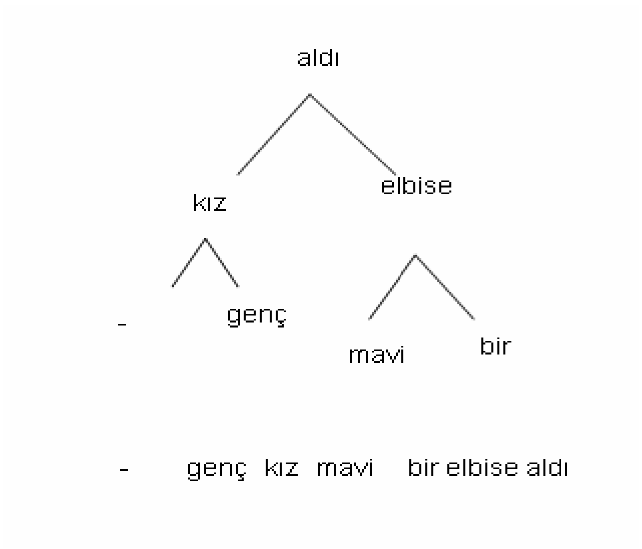


Figure 4.8 Alignment of SSTC

CHAPTER V

ORHUN: AN EXAMPLE-BASED MACHINE TRANSLATION SYSTEM BASED ON THE SYNCHRONOUS SSTC

In this chapter, the design of the ORHUN system will be described. In this system, translation from English to Turkish is performed using synchronous SSTC annotation schema as in the EBMT. The system has been named as ORHUN which is the first written monument in Turkish. The design of the ORHUN system is shown in Figure 5.1.

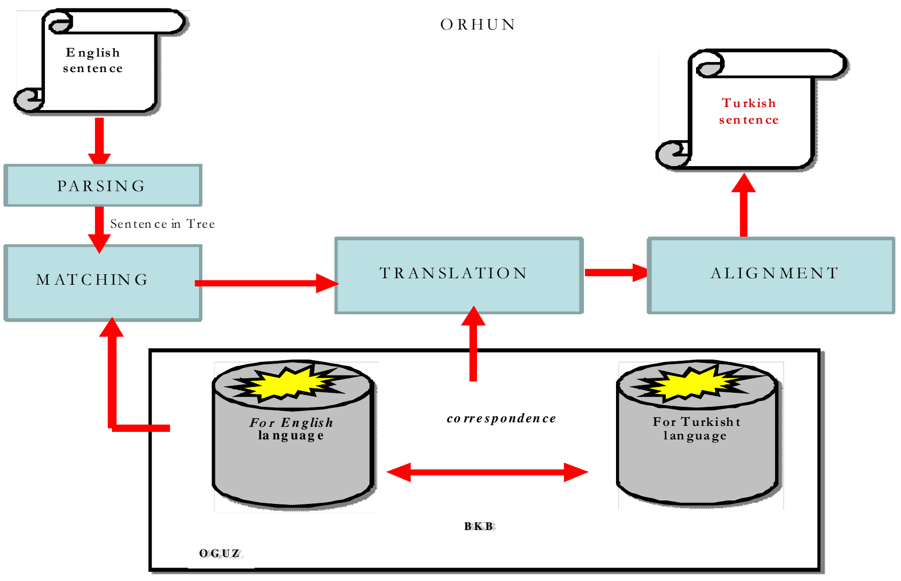


Figure 5.1 Design of the ORHUN system

The system has three main components which are parsing, matching and translation/alignment. Firstly, the parsing component parses a given sentence and forms a parameter in the form of a tree which includes the information about the given sentence. In the matching module, the separate subtrees are formed from the tree parameter, and these subtrees are compared against the subtrees of the other English sentence trees retrieved from the database. These subtrees are compared by using the top down search method. If the system finds equal subtrees, the Turkish subtree which corresponds to the matched English subtree is retrieved. All of the subtree matching process finishes but some subtrees have not been translated then the system retrieves another English sentence tree from the database. The system requires the usage of large corpora; consequently the storage mechanism becomes especially important for the system. Before giving the sample run for the components of the system, example sets of storage will be describe in the following section.

5.1 System Domain

As the domain of the MT system, MS Office Help documents have been selected. As EBMT system requires parallel corpora, MS Office Help documents, which includes one English sentence corresponds to one Turkish sentence, proves a suitable domain. However for the system to operate, both English and Turkish parsers were required. Yet, the number of parsers developed for the Turkish language that were available to be used by other systems was minimal. There are a lot of works about Turkish NLP at Boğaziçi University [45], Bilkent University [46], METU [47], and Istanbul Teknik University [48], and Turkish parsers have been developed, however, either there was no access to them, or they did not produce high-quality results for this working. Finding the word types and grammatical structure of a Turkish sentence to create the Turkish tree structure created a major problem with these parsers. For example, Zemberek [49], and the parser developed by Bogaziçi University [45] and METU [47] were tried, but access was not allowed.

As mentioned before, MS Office help documents were used as the parallel corpora. Figure 5.2 shows some English sentences and their corresponding Turkish translations found in the help documents. As seen in the examples in Figure 5.2, there are a lot of similar sentences in the help documents.

In the Style type box, click Character.	Stil türü kutusunda Karakter'i tıklatın.
In the Style type box, click List.	Stil türü kutusunda Liste'yi tıklatın.
In the Style type box,click Paragraph.	Stil türü kutusunda Paragraf'ı tıklatın.
In the Style type box, click Table.	Stil türü kutusunda Tablo'yu tıklatın.

Figure 5.2 Examples of MS Office help document sentences

5.1.1 Storage of Examples

The system uses SSTC annotation schema for the storage of the sentences, therefore a table structure is required which can include a tree structure. For parsing an input sentence, the Stanford Parser [50] is used and the resulting tree is stored in the table. The details about the Stanford Parser [50] are given in the next section.

5.2 System Components

ORHUN has three main components. Parsing is the first step where a source sentence is parsed and outputted as a tree. The next step is matching where the tree that includes the information about the source sentence is compared against the corpus sentences to try to find a similar sentence in the database. If a similar sentence is found, the corresponding Turkish sentence is taken from the database. In order to perform these steps, system must parse the source sentence correctly.

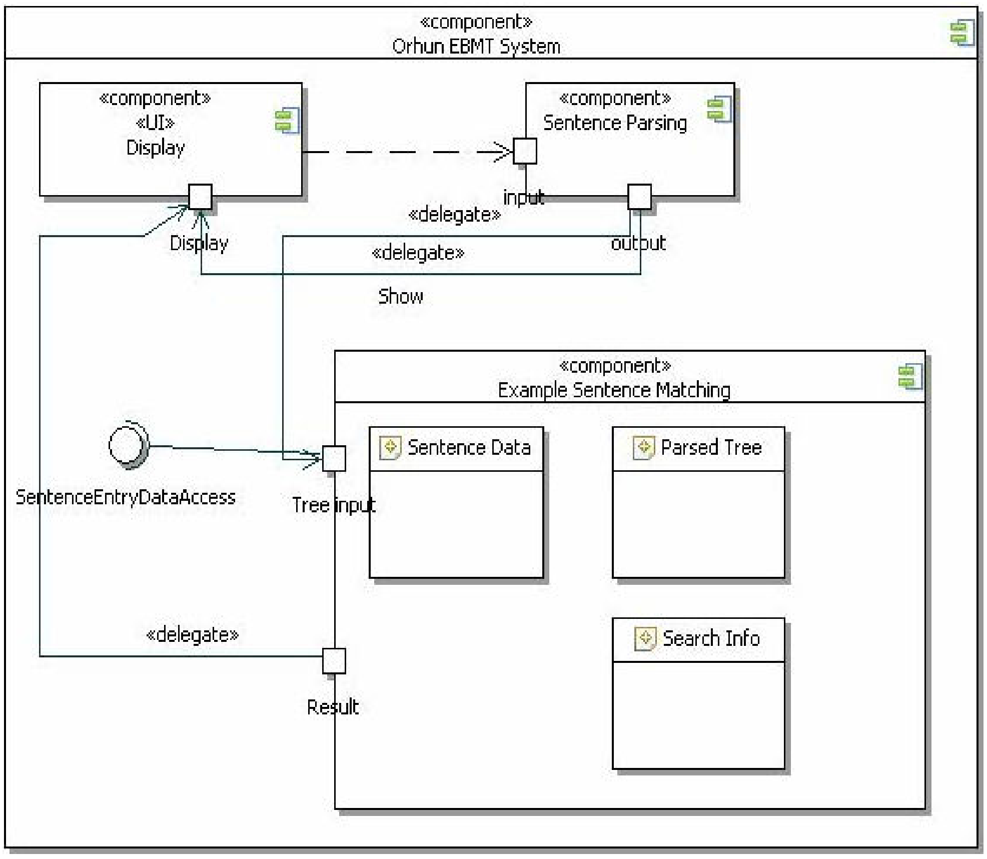


Figure 5.3 Design of the ORHUN system components

5.2.1 Parsing Component

A natural language parser is an application that analyzes the grammatical structure of sentences. For example; given a sentence, the parser decides on the categories (e.g. noun, verb, adjective, etc.) of all the words, and the roles (e.g. subject, object, etc.) they play in the sentence. There has been more academic research about parsing in the English language than in any other language. In comparison, the research on developing a parser for the Turkish language is one of the most important problems in MT research dealing with the Turkish language.

Probabilistic parsers use knowledge of language gained from hand-parsed sentences to try to produce the most likely analysis of new sentences. This approach may result some mistakes, but in general gets close to the correct result. For parsing the English sentences, Stanford Lexicalized Parser package is used [50]. The Stanford Lexicalized Parser [50] is a statistical NLP parser developed at Stanford University. The corpus that is used by the Stanford Parser is like a corpus that used the Penn Treebank style used bye The University of Pennsylvania [50]. It uses probabilistic context free grammar (PCFG) parser which outperforms standard lexicalized models used for English. These grammar rules are implemented in a file named englishPCFG.txt.

The Stanford Lexicalized Parser package is a Java implementation of probabilistic natural language parsers. The parser supports highly optimized PCFG and dependency parsers, and a lexicalized PCFG parser. The lexicalized probabilistic parser implements a factored product model which uses the A* algorithm.

The version of the parser which was used by ORHUN requires at least JDK 1.4. The parser also requires plenty of memory (a minimum of 100Mb to run as a PCFG parser on sentences up to 40 words in length; typically around 500Mb of memory to be able to parse similarly long typical-of-newswire sentences using the factored model).

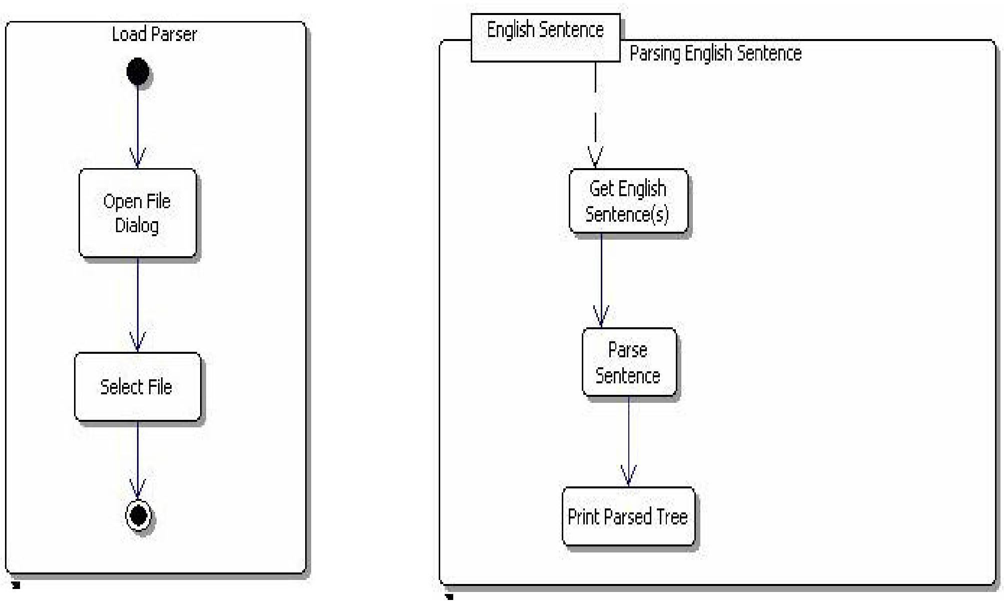


Figure 5.4 Parsing Component Structure

Figure 5.4 shows the Parsing Component Structure. First of all, the system selects a PCFG file for English language and writes a sentence for parsing and then the parser provides grammatical relations output as well as phrase structure trees for this sentence. In Figure 5.5, the parse of the sentence “She bought a blue dress” by the Stanford Parser is shown.

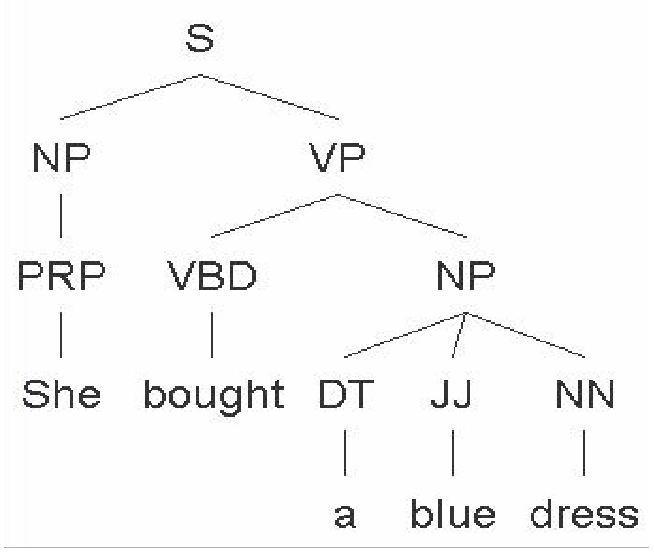


Figure 5.5 Example of a tree built by Stanford Parser

ORHUN implements the SSTC method to translate a given sentence, which requires a tree structure as the storage mechanism. To implement this method, a tree is generated by the Stanford Lexicalized Parser which is stored in the tree database. Figure 5.6 shows the main tables in the database. In *EnglishSentence* table, SentenceId is a unique number for a sentence. *LexId* is an id to show the part of the sentence. *LexType* is a string that shows type of word. *ParentLexId* is an id that shows parent’s Id and *Word* shows a word. If Word is not null this shows that it is a leaf in the tree.

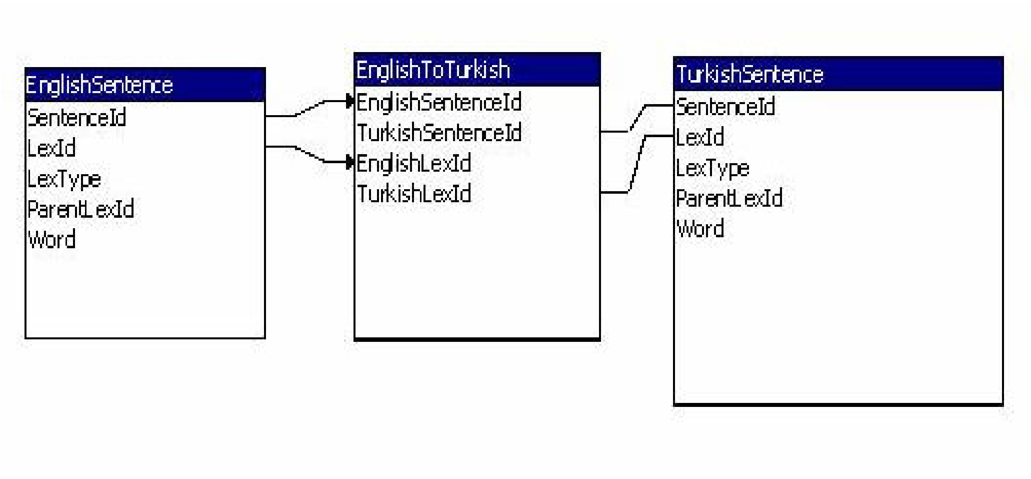
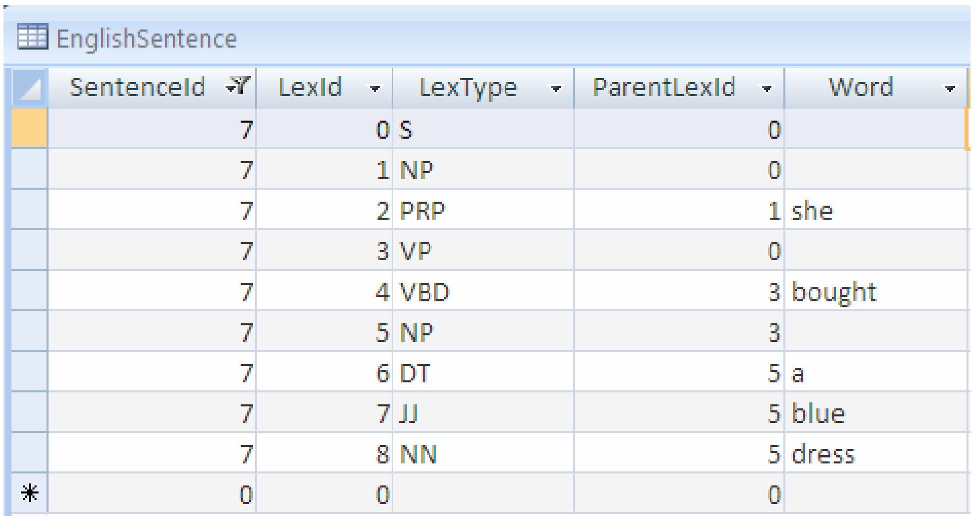


Figure 5.6 Table structure of the Database

In Figure 5.7, the storage of the example sentence "She bought a blue dress" in the *EnglishSentence* table is shown.



EnglishSentence

	SentenceId	LexId	LexType	ParentLexId	Word
	7	0	S	0	
	7	1	NP	0	
	7	2	PRP	1	she
	7	3	VP	0	
	7	4	VBD	3	bought
	7	5	NP	3	
	7	6	DT	5	a
	7	7	JJ	5	blue
	7	8	NN	5	dress
*	0	0		0	

Figure 5.7 Example of "She bought a blue dress" in the EnglishSentence

5.2.2 Matching Component

This step is the most important step for the translation systems that use EBMT. Firstly, the system takes a tree of an input sentence from the parser. This tree, called as "source tree" (ST), is separated to subtrees by the system. Since one sentence can simply include subject, object and verb; the system has to search component by component. Like ST, the system takes a tree from the database called as "database tree" (DT). the system must separate the DT to subtrees for comparison to ST's subtrees. Figure 5.8 shows the matching component structure.

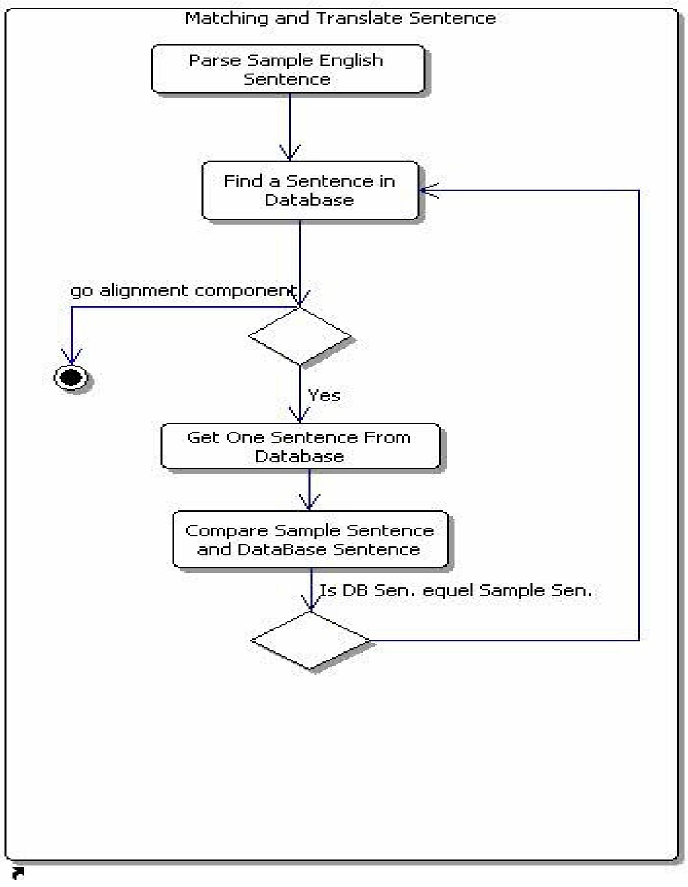

Matching and Translate Sentence
Parse Sample English Sentence
Find a Sentence in Database
go alignment component
Yes
Get One Sentence From Database
Compare Sample Sentence and DataBase Sentence
Is DB Sen. equel Sample Sen.

Figure 5.8 Matching Component

After the separation process, the system takes one tree referred as *st1* which is a subtree of ST and compares it to a tree *dt1* that is a subtree of DT. The system compares *st1* and *dt1* node by node. If the two trees have equal nodes, this shows that these subtrees (part of sentence) have the same grammatical structure. If two trees do not have equal nodes, the system takes a new subtree from the DT.

If the system finds that *st1* and *dt1* are equal, the system looks at their leaves that include words in the sentence. Thus, the system finds a word in the database that has the same grammatical task as the ST word. If the system finds all of leaves in the *st1* equal to *dt1* this shows us that the system can translates *st1* to Turkish. The system takes a Turkish sentence or part of a sentence from the database that corresponds to *dt1* and the system adds a character (**) in the subtree root which shows us that this sentence is translated.

Next, the system takes another subtree from the ST and tries to find an equal subtree in the DT. If ST is finished and the system has a subtree that does not find equal subtrees in the DT yet, the system takes another DT in the database. Consequently, the system looks for all database sentences. If the system finds equal subtrees for the combination ST translation ends in success.

5.2.3 Alignment Component

The Turkish grammatical structure is different from the English grammatical structure. For example, English sentence structure is SVO (Subject, Verb, Object). However Turkish sentence structure is SOV (Subject, Object, Verb). This is a problem for translating English to Turkish. Figure 5.9 explains this problem.

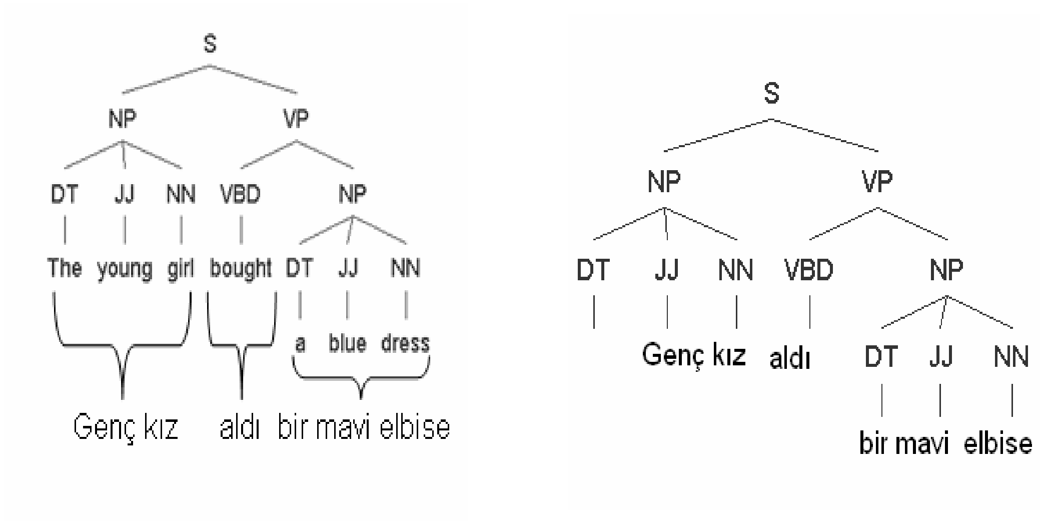


Figure 5.9 An example of result of matching

For the ORHUN system, MS Office Help Document, in which sentences nearly have the same grammatical structures, is used. Only the sentences that are built up with Simple Present Tense are used in corpus. Hereby, the domain is limited out to simple sentences by using present tense. With a good Turkish parser, however, this problem can be solved, which will then increase the variety of the sentences that can be translated.

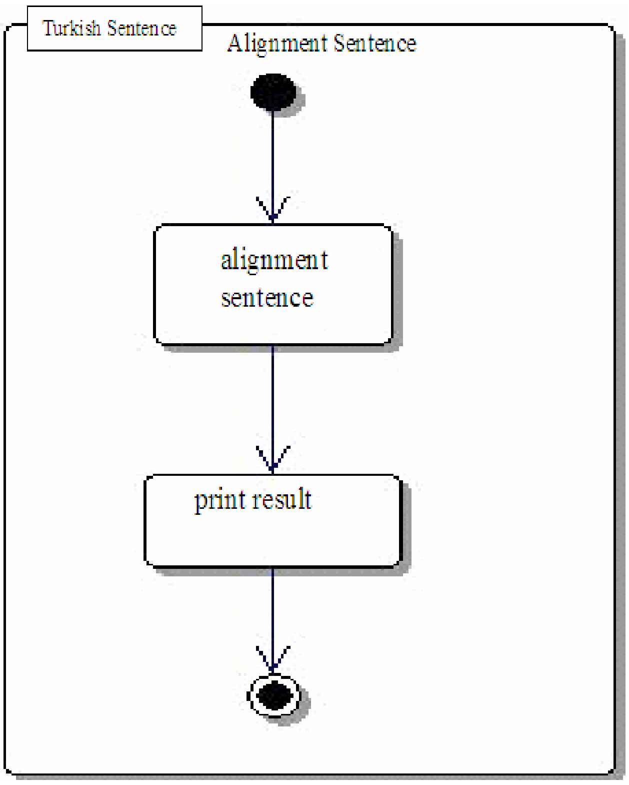


Figure 5.10 Alignment Component

Figure 5.10 shows the alignment component. ORHUN uses basic alignment rules. For example, in Turkish, Object comes before the Verb, whereas in English, the Verb comes before the Object. Thus, the corresponding tree nodes for Verb and Object need to be interchanged. In the future, these rules will be extended.

5.3 Implementation of ORHUN

The system is developed with the Java programming language (JDK 1.5) using Borland Jbuilder 2006 integrated development environment. The English and Turkish example sentence sets have been stored in a sample database developed in Ms Access which has been named as OGUZ.

ORHUN uses Stanford Parser for parsing the English sentence, so the system uses Stanford Parser tree structure. First of all, it is better to explain this tree structure.

5.3.1 Class Tree

ORHUN is based on a tree structure. The system implemented all tree applications in a tree class. For example, a tree is created and a child is added or trees are compared whether they are equal. Stanford Parser uses A(*) algorithm and the system use top down search method for searching in the tree. Appendix B shows some methods of Class Tree.

5.3.2 Class SentenceDataAccess

This class includes the methods for all of the database actions. First of all *connect()* method makes a connection by OGUZ database, and this class uses SentenceEntry class that denotes sentence structure. Some of the methods in this class will be explained shortly.

public SentenceEntry findSentence(int sentenceId,int lexId): takes two integers as parameters, finds a part of a sentence from the *EnglishSentence* table from the database with a given *sentenceId* and *lexId*. This function returns a *SentenceEntry*.

public SentenceEntry findTurkishSentence(int sentenceId,int lexId): this method is similar to *findSentence* but this method takes data from the *TurkishSentence* table in the database.

public int getMaxSentId(): gives maximum *sentenceId* from the *EnglishSentence* table.
public boolean newSentence(SentenceEntry sentence): inserts an English sentence to the

EnglishSentence table in the database. Before the insert, it calculates (maximum SentenceId + 1) for primary key of *EnglishSentence* table.

public boolean saveSentence(SentenceEntry sentence): takes type of *SentenceEntry* as a parameter and then it updates *EnglishSentence*.

public boolean deleteSentence(SentenceEntry sentence): deletes an English sentence from the *EnglishSentence* table in the database.

public ArrayList getSentenceIdArrayList(): gives a list that includes all *SentenceId*s. The system may search all sentences during a matching process. In matching phase the system takes a sentence and compares it to source sentence and then if system does not match, the system must retrieve a new sentence from the database. So the system uses *getSentenceIdArrayList* to take all *sentenceIds* from the *EnglishSentence* table.

public Tree getSentenceTree(int sentenceId): takes an integer as a parameter and takes a sentence from *EnglishSentence* table given by sentence's id and returns it.

public *Tree getTurkishSentenceTree(int sentenceId)*: like *getSentenceTree* method, it takes an integer as a *sentenceId* and takes a sentence from the *TurkishSentence* as a tree and returns it.

public void close(): the connection to the Oguz database before the system is closed.

5.3.3 Class EBMT Frame

This class deals the screen item positions and controls workflow. This class uses *EnglishPanel* and *TurPanel* classes.

public static void main(String args[]): Our program starts with this method. This method calls *EBMTFrame* that includes menu for some operations as well as the *EngPanel* and

TurPanel. These panels will be explained below. The components of menu will be explained shortly.

private void jbInit(): this method includes screen item property and its given initial knowledge of fields that includes menu and panels.

Parser (Load File, Load Parser, Exit)

English (Save Database, Get EnglishSentenceTree from DB)
    Turkish(Save Database)
    EBMT(searchDatabase)

private void exitForm: closes the program.

5.3.3.1 Class EngPanel

This class has a panel which includes the English sentence. User enters an English sentence which is initially parsed by the system and at the bottom half of the screen the tree that shows the sentence structure is outputted. Other important methods in the *EngPanel* are described below;

public void loadFile(): User can write one sentence or select a file that includes several sentences to be parsed.

public void loadParser() : The system must use a PCFG file for parsing. User can choose one of the *englishPCFG.txt* or *englishPCFG.ser.gz* files. Stanford Parser uses these files for parsing.

public void parse(): This method parses the sentence and shows the sentence on the screen as a tree.

5.3.3.2 Class TurPanel

During the matching phase, if the system finds one (or a lot of) sentence in the database that matches to the source sentence or part of source sentence, the system shows this sentence or sentences and matching part in the *TurPanel*. If the system can make a translation from source sentence to Turkish sentence, it shows translation sentence in the *TurPanel* again.

5.3.4 Class EBMT

This class performs all of the translation actions. The method details are explained below:

public static Collection subTrees (Tree t, Collection n): the system makes a search in the tree during the matching phase. It uses top down search. Since the system can find the same part of source sentence in any part of database sentence it sometimes searches all tree, or sometimes it searches only one part of the tree. The system takes all subtrees of given tree in a collection and returns it.

public static List subTreesList (Tree t): calls *subTree* method, takes a collection and finally returns a list of this collection.

public List compareTwoTree (List tList1, List tList2): during the matching phase, it compares two tree lists member by member. This method has a kind of control in order to shorten the matching time. For example, if a subtree is found during the matching the source sentence, the system adds a (**) or it searches the subtree, but if the system cannot find similar part of sentence in the database, it adds these subtrees in to the list that is searched before.

public int compareTo (Tree t1, Tree t2): compares two trees leaf to node. This method is a recursive function. It calls itself that has *subt1*, and *subt2* as a parameter. *subt1* is a child of *t1* and *subt2* is child of *t2*, so it compares all nodes of *t1* and *t2*. It returns 0 if *t1* and *t2* are equal.

public int finishSearch (Tree t): controls whether all nodes have been found or not. The system adds (**) in the node of source sentence tree if the system can makes a translation perfectly. This means, the system found an English sentence with the same structure and word in the database. This method is a recursive function. It calls itself so the system can understand if the source sentence is translated.

public Tree searchSentence (Integer stSentID): This method takes one integer that uses source of *Sentence Tree* and constructs a tree *st* and then builds *stList* with *subTreesList()* that includes *st*'s subtree. Next it uses *getSentenceIdArrayList()* that constitutes the list that included all *sentenceIds* in the *EnglishSentence* table.

Iteratively, the system takes one sentence in a tree with *sentenceId* in the *SentenceIdArrayList*, and the system builds *dtList*. The system uses *compareTwoTree* with *stList* and *dtList*. If the system finds equal subtrees (part of sentence) in the *compareTwoTree*, system changes this part with corresponding Turkish part of sentence by *getTurkishSentenceTree*. The system appends two stars (**) if a translation has perfect meaning structure and leaves are equal, or the system adds one star (*) if only a leaf is equal. If *finishSearch* returns a 0, then the system takes a new sentence from the database with a new *sentenceId in SentenceIdArrayList*. This operation continues until *finishSearch* returns 1 (translation is complete) or when *SentenceIdArrayList* is finished.

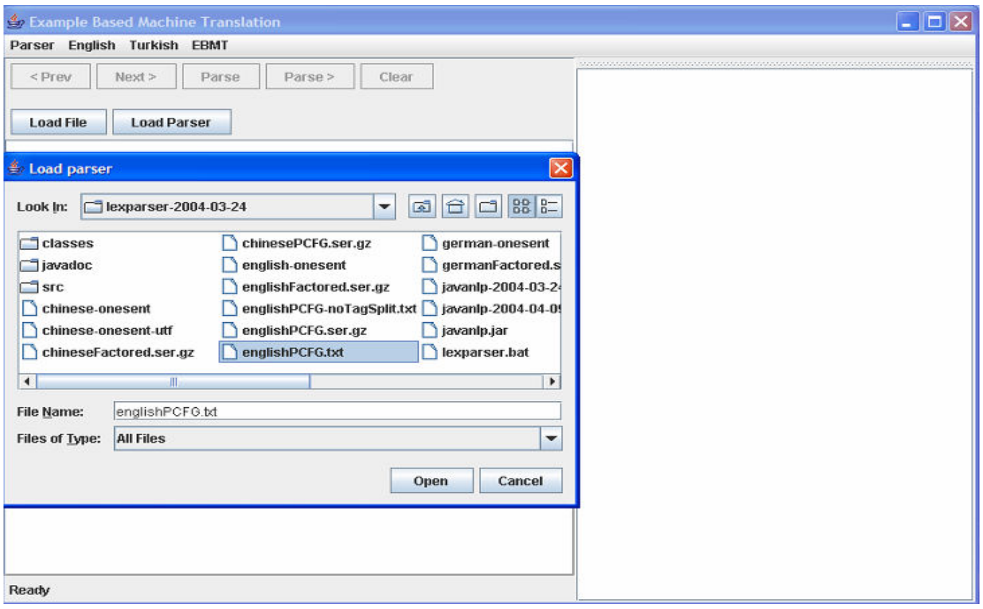
public Tree alignmentSentence(Tree t): Turkish sentence structure is different from English sentence structure. Here, the system creates a tree which is suitable to the English sentence structure. System must reorganize this tree for Turkish sentence structure. For this purpose *S-SSTC* is used in ORHUN suitable to the Turkish sentence during the translations. The system uses a basic rule for this process which will be explained later with an example.

public void translateEnglishToTurkish(Integer stSentID) : The system starts to translate with *translateEnglishToTurkish* method. It calls *searchSentence* for mathing source sentence to the database sentence and then *searchSentence* checks whether the translation is a success. If translation is a success, *searchSentence* finishes the search and returns a tree that includes Turkish sentence. If this tree is not null, this means that translation is successful and it calls *alignmentSentence* method.

In the next section ORHUN system will be explained with an example, to show how the system is executed and how the PCFG file for the parser is selected.

5.4 Interface of ORHUN

Firstly, user selects a PCFG file with “Load Parser” button and the ORHUN system loads selected file for parsing an English sentence. Figure 5.11 and 5.12 show this action.

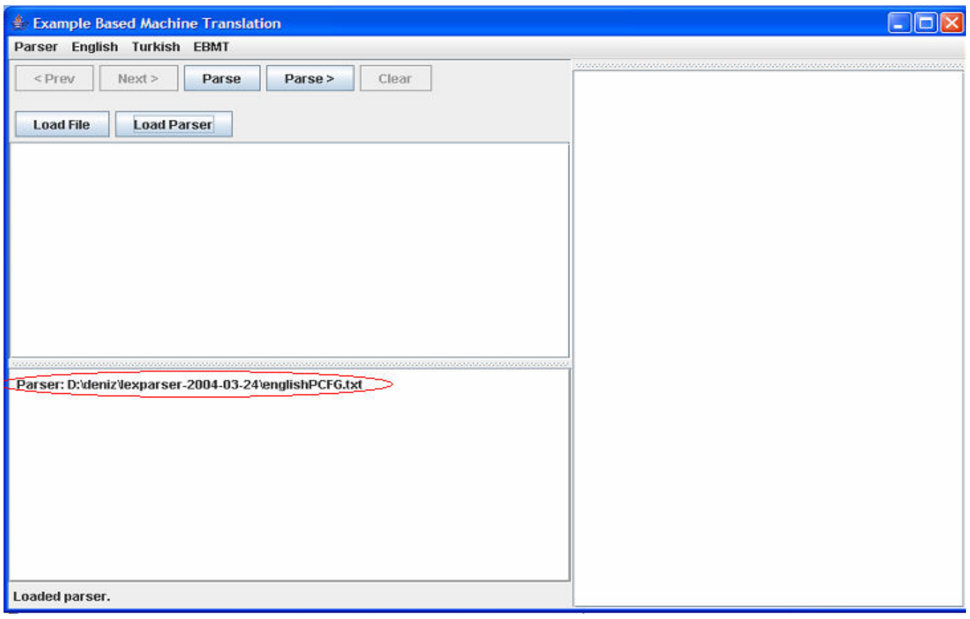


Look In: lexparser-2004-03-24

File Name: englishPCFG.txt

Files of Type: All Files

Figure 5.11 Select PCFG file (englishPCFG.txt) for ORHUN system

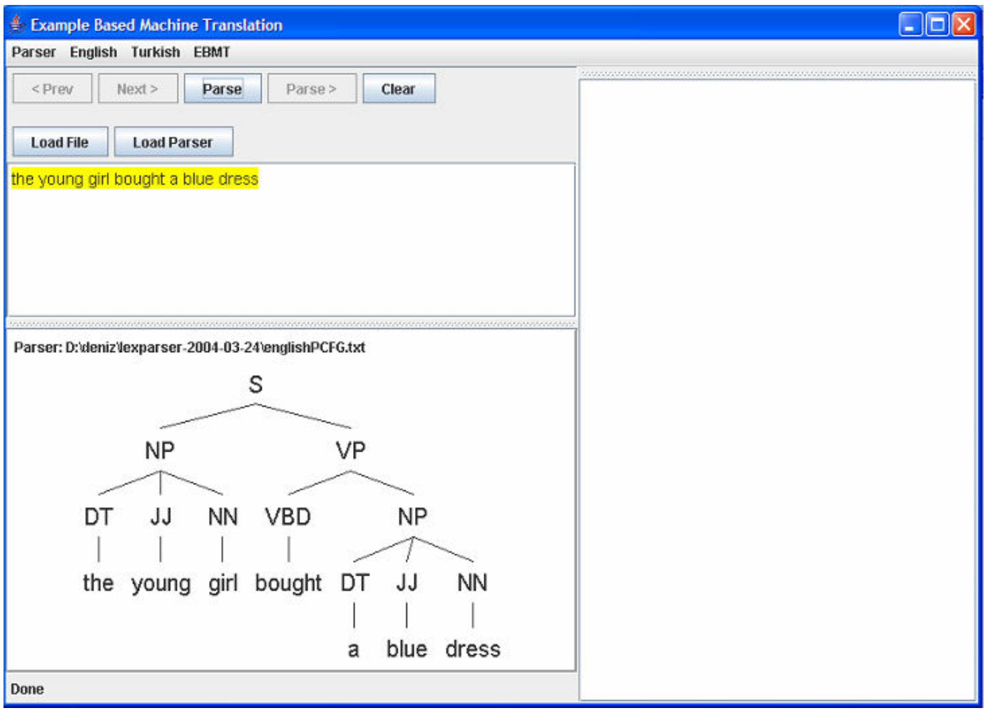

Example Based Machine Translation
Parser English Turkish EBMT
< Prev
Next >
Parse
Parse >
Clear
Load File
Load Parser

Parser: D:\deniz\lexparser-2004-03-24\englishPCFG.txt

Loaded parser.

Figure 5.12 System loaded englishPCFG.txt as a parser file

Now the system starts to translate the English sentence to the Turkish one. Firstly, the system writes a sentence and parses it as shown in Figure 5.13.



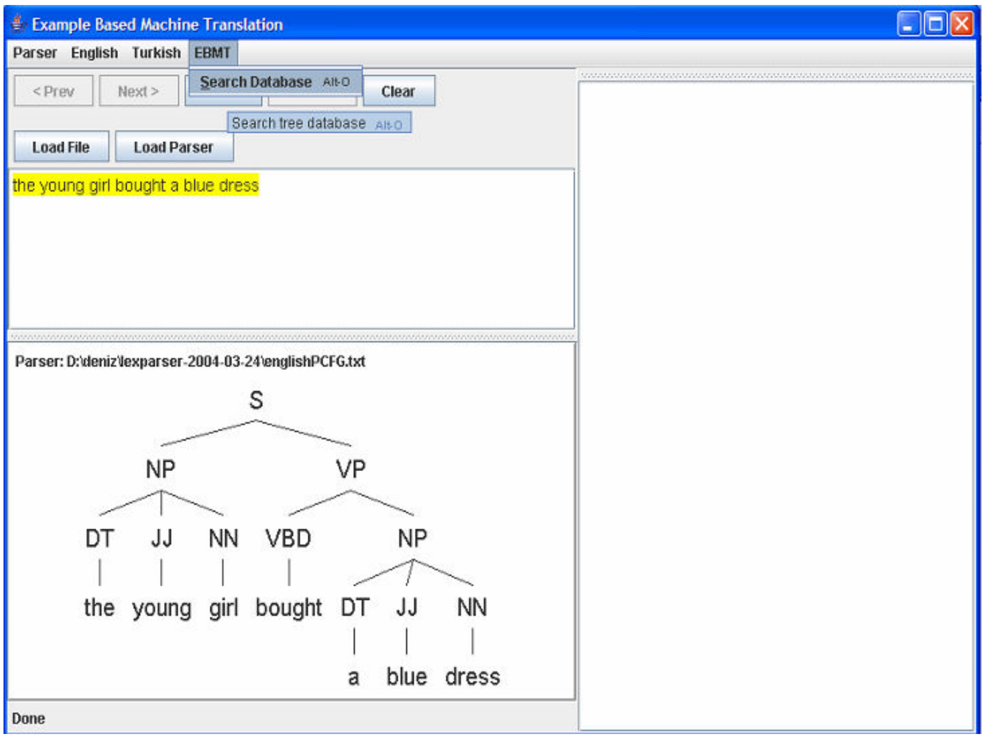
the young girl bought a blue dress

Parser: D:\deniz\lexparser-2004-03-24\englishPCFG.txt

Done

Figure 5.13 “the young girl bought a blue dress” source sentence is parsed

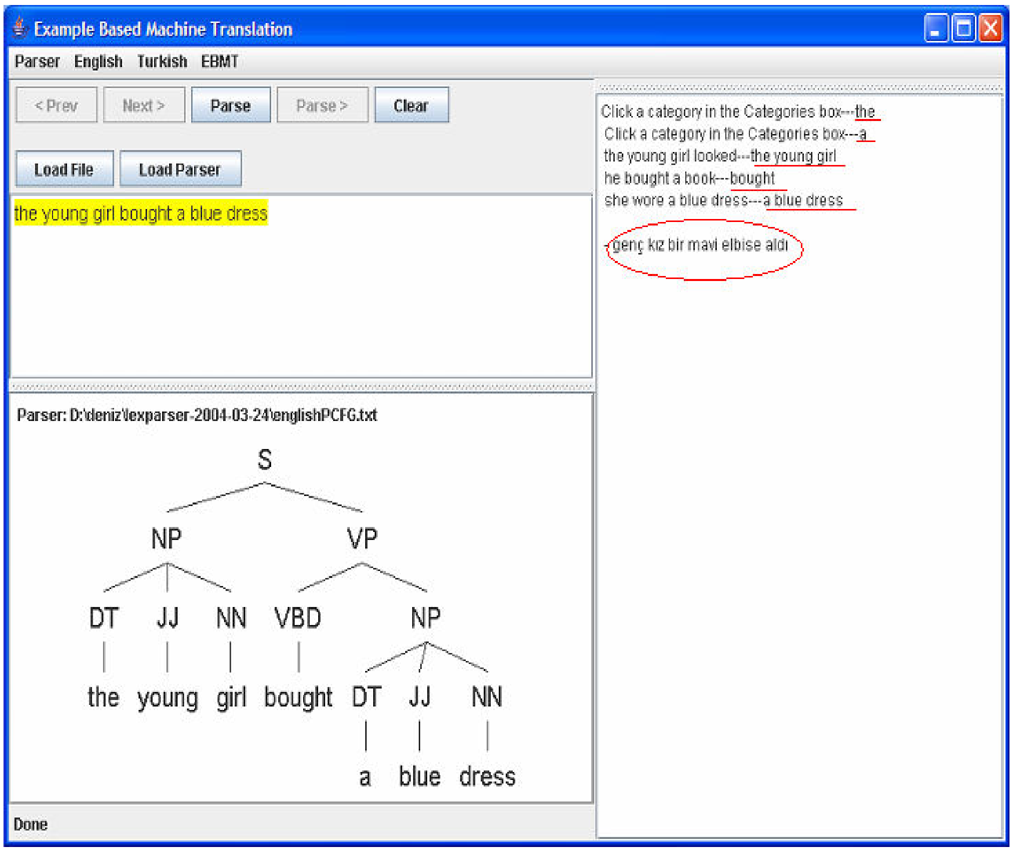
Then, user selects the *searchSentence* option from the menu. Accordingly, the system selects English sentence from the database and compares it to the source sentence. If the system finds a similar sentence or part of sentence, it takes the Turkish sentence or part of the sentence that corresponds to the found sentence. These steps are shown in Figure 5.14 and 5.15.



the young girl bought a blue dress

Parser: D:\deniz\lexparser-2004-03-24\englishPCFG.txt

Figure 5.14. Start to searchDatabase on the menu


Example Based Machine Translation
Parser English Turkish EBMT
< Prev
Next >
Parse
Parse >
Clear
Load File
Load Parser
Done

the young girl bought a blue dress

Parser: D:\deniz\lexparser-2004-03-24\englishPCFG.txt

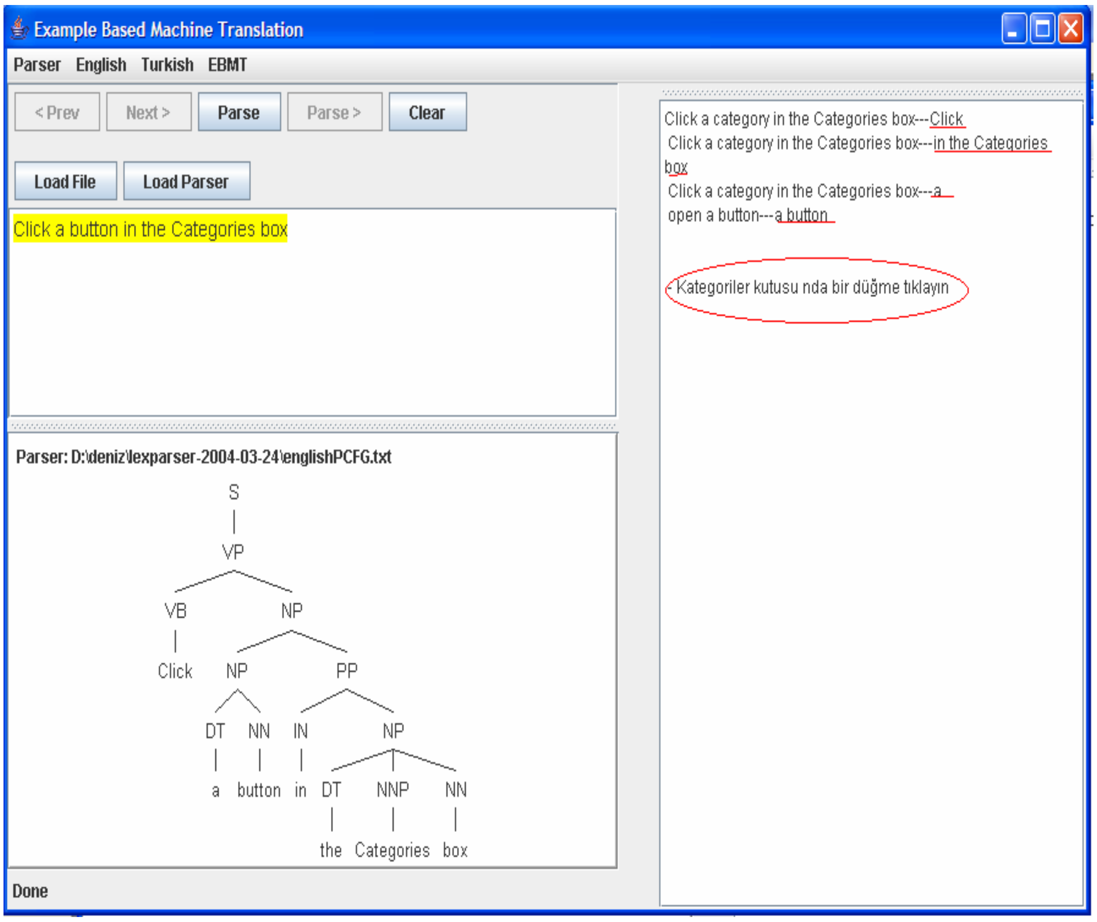
Click a category in the Categories box---the
Click a category in the Categories box---a
the young girl looked---the young girl
he bought a book---bought
she wore a blue dress---a blue dress

-genç kız bir mavi elbise aldı

Figure 5.15 Sample 1: Result of English sentence translation to Turkish sentence

Figure 5.15 shows that “The young girl bought a blue dress” corresponds to “Genç kız bir mavi elbise aldı”. During the translation process, the system has found 5 similar sentences and selected the most suitable one.

Figure 5.16 shows another example of the system. Here, “Click a button in the Categories box” corresponds to “Kategoriler kutusunda bir düğme tıklayın”. During the translation process, the system has found 5 similar sentences and selected the most suitable one.


Example Based Machine Translation
Parser English Turkish EBMT
< Prev
Next >
Parse
Parse >
Clear
Load File
Load Parser
Done

Click a button in the Categories box

Parser: D:\deniz\lexparser-2004-03-24\englishPCFG.txt

```
S
  VP
    VB Click
    NP
      NP
        DT a
        NN button
      PP
        IN in
        NP
          DT the
          NNP Categories
          NN box
```

```
Click a category in the Categories box---Click
Click a category in the Categories box---in the Categories box
Click a category in the Categories box---a
open a button---a button

Kategoriler kutusu nda bir düğme tıklayın
```

Figure 5.16 Sample 2: Result of English sentence translate to Turkish sentence

5.5 Tests and Evaluation

In this works 139 different sentences is inserted into the *EnglishSentence* table and the corresponding sentences in the *TurkishSentence* table. Appendix A includes our corpus that is built similar sentences. Number of perfect translations is satisfactory but generally it is thought that this success depends on the corpus. It is expected that, if the number of examples are increased, the success ratio will also increase. Additionally, if ORHUN can't make a perfect translation, it will translate the source sentence word by word that corpus has those words. Thus, it will be understood what the sentence means, in anyway.

CHAPTER VI

CONCLUSION AND FUTURE WORK

In this thesis, Synchronous Structured String-Tree Correspondence (S-SSTC) system has been developed to perform EBMT between English and Turkish languages. EBMT methodology uses sentences or parts of sentences that were translated beforehand which are stored as a corpus of translated sentences. The developed software uses SSTC for the storage of the sentence information.

The system uses S-SSTC for the translation between the source language (English) and the target language (Turkish), since the structure tree used by this methodology provides flexibility for the Turkish grammar structure. In the translation phase, ORHUN aims to produce readable translations without emphasizing the grammatical differences between the two languages. Using simple rules for alignment, the system can produce perfect translations. In the literature survey, only one working system between English to Malay languages was found which used the same methodology; consequently this methodology has been applied for the first time between English and Turkish languages.

One of the most important components of ORHUN is the corpus. If the corpus is built with similar sentences in source language, system success rate decreases dramatically. If the corpus is limited to a certain topic, the system provides more effective results. For example, a corpus can be created which only relates with medicine, or computer programming, as the sentences in the same topic generally have the same structure and terminology. Thus, given a corpus on medicine, if there is a sentence about one illness, there probably will be a similar sentence about another illness. Consequently, developing a corpus about a topic will enable us to translate any manual or text book related with that topic. In addition to this, The system performance can improve by a lot of database techniques. For example if the system uses index on the database tables, system performance will be increase.

The most difficult problem encountered in this thesis is finding a Turkish parser. After analyzing all the Turkish parsers available, suitable application cannot be found for our software. Therefore, the Turkish sentence trees used by ORHUN had to be created manually. For this reason, the example translations provided in the corpus of the system were minimal. In the future work, the corpus can be enlarged to provide other examples, and other corpuses can be built on other topics. If a powerful Turkish parser is built, the problem of developing large English-Turkish example corpuses will be easily solved. The ORHUN EBMT system can also be extended to include other grammatical structures such as compound sentences, tenses, etc. Different EBMT methodologies could be tested on the same corpus to analyze which methodology provides the best result in English and Turkish translation. Finally as a future work, languages other than English as the source language could be chosen to provide translation into Turkish.

Nowadays, especially with the enormous impact of the Internet, a large amount of translation from English to Turkish is needed. The quality of the translation on these documents is not very important, yet the time required for the translations has to be minimal. Providing easy, quick and readable translations on the Internet will especially be helpful to Turkish speaking users of the Internet to access any document in English. The ORHUN system developed for this thesis provides an initial step to fast and easy translation between English and Turkish.

REFERENCES

[1] Doug Arnold: Machine Translation: an Introductory Guide Blackwells-NCC, London, 1994,

[2] http://www.eamt.org/

[3] Http://en.wikipedia.org/wiki/Machine_translation

[4] John Hutchins Latest Developments in Machine Translation Technology: Beginning a New Era in MT research n: The Fourth Machine Translation Summit: MT Summit IV. Proceedings: International cooperation for global communication, July 20-22, 1993, Kobe, Japan

[5] W.John Hutchins Machine Translation: A Brief History - Concise history of the language sciences: from the Sumerians to the cognitivists. Ed. E.F.K.Koerner and R.E.Asher (Pergamon, 1995), pp.431-445

[6] Anna Sågvall Hein Machine Translation www.gslt.hum.gu.se/~masaers/MT06/MT06_slides_Anna.ppt

[7] John Hutchins Machine translation and computer-based translation tools: what's available and how it's used- A new spectrum of translation studies, ed. José Maria Bravo (Valladolid: University of Valladolid, 2004),

[8] http://www.globalsecurity.org/intell/systems/mt-techniques.htm

[9] Francis Bond Toward a Science of Machine Translation- Proceedings of the MT Roadmap Workshop at TMI-2002, Keihanna, Japan

[10] Davide Turcato, Fred Popowich What is Example-Based Machine Translation- In M. Carl and A.Way, editors, Recent Advances in Example-Based Machine Translation, pages 59–83, Dordrecht, The Netherlands, 2003.

[11] Harold Somers (1999). Review article: Example-based Machine Translation, Machine Translation, 14: 113-157.

[12] Harold Somers 2003 An Overview of EBMT In M. Carl. and A. Way. (eds.) Recent Advances in Example-based Machine Translation,Kluwer Academic Publishers, Dordrecht, The Netherlands, pp.3-57.

[13] Kevin Duh Example-Based Machine Translation http://ssli.ee.washington.edu/people/duh/

[14] Watanabe, H. A Similarity-Driven Transfer System- Coling (1992), 770–776

[15] Al-Adhaileh, M. H. And Tang E. K. Example-Based Machine Translation Based on the Synchronous SSTC Annotation Schema- Machine Translation Summit VII, Singapore, 244–249.,199

[16] Brown, R. D. Adding Linguistic Knowledge to a Lexical Example-based Translation System - TMI (1999), 22–32.

[17] Weaver, A. 1988. Two Aspects of Interactive Machine Translation. In M. Vasconcellos (ed.) Technology as Translation Strategy, 116–123, Binghamton, NY: State University of New York at Binghamton (SUNY).

[18] Nagao, M. 1984. A Framework of a Mechanical Translation between Japanese and English by Analogy Principle. In A. Elithorn and R. Banerji (eds) Artificial and Human Intelligence, 173–180, Amsterdam: North-Holland.

[19] Carroll, J. J. 1990. Repetitions Processing Using a Metric Space and the Angle of Similarity. Report No. 90/3, Centre for Computational Linguistics, UMIST, Manchester, England.

[20] Maruyama, H. And H. Watanabe. 1992. Tree Cover Search Algorithm for Example-Based Translation. TMI (1992), 173–184.

[21] Matsumoto, Y., H. Ishimoto and T. Utsuro. 1993. Structural Matching of Parallel Texts. 31st Annual Meeting of the Association for Computational Linguistics, Columbus, Ohio, 23–30.

[22] Watanabe, H. 1995. A Model of a Bi-Directional Transfer Mechanism Using Rule Combinations. Machine Translation 10, 269–291.

[23] Al-Adhaileh, M. H. And Tang E. K. 1998. A Flexible Example-Based Parser Based on the SSTC. Coling-ACL (1998), 687–693.

[24] Utsuro, T. K. Uchimoto, M. Matsumoto and M. Nagao. 1994. Thesaurus-Based Efficient Example Retrieval by Generating Retrieval Queries from Similarities. Coling (1994), 1044–1048.

[25] Nirenburg, S., C. Domashnev and D. J. Grannes. 1993. Two Approaches to Matching in Example-Based Machine Translation. TMI (1993), 47–57.

[26] Sato, S. 1995. MBT2: A Method for Combining Fragments of Examples in Example-Based Machine Translation. Artificial Intelligence 75, 31–49.

[27] Watanabe, H. 1995. A Model of a Bi-Directional Transfer Mechanism Using Rule Combinations. Machine Translation 10, 269–291.

[28] Lepage, Y. 1994. Texts and Structures – Pattern-matching
and Distances, ATR report TR-IT-0049, Kyoto, Japan

[29] Kaji, H,Y. Kida and Y. Morimoto. 1992. Learning Translation Templates from Bilingual Text. Coling (1992), 672–678.

[30] Sumita, E., H. Iida and H. Kohyama. 1990. Translating with Examples: A New Approach to Machine Translation. TMI (1990), 203–212.

[31] Somers, H., I. Mclean and D. Jones. 1994. Experiments in Multilingual Example-Based Generation. CSNLP 1994: 3rd Conference on the Cognitive Science of Natural Language Processing, Dublin, Ireland

[32] Brown, R. D. 1997. Automated Dictionary Extraction for “Knowledge-Free” Example-Based Translation. TMI (1997), 111–118.

[33] Veale,T. and A.Way. 1997. Gaijin: A Bootstrapping Approach to Example Based Machine Translation. International Conference, Recent Advances in Natural Language Processing, Tzigov Chark, Bulgaria, 239–244.

[34] Collins, B. 1998. Example-Based Machine Translation: An Adaptation-Guided Retrieval Approach. PhD thesis, Trinity College, Dublin

[35] Collins, B. and P. Cunningham. 1997. Adaptation Guided Retrieval: Approaching EBMT with Caution. TMI (1997), 119–126.

[36] Smyth, B. and M. T. Keane 1993. Retrieving Adaptable Cases. In I. Smith and B. Faltings (eds), Topics in Case-Based Reasoning: Proceedings of the First European Workshop on Case-Based Reasoning, EWCBR-93, Vol. 1, 76–82, Berlin: Springer.

[37] Grefenstette, G. 1999. The World Wide Web as a Resource for Example-Based Machine Translation Tasks. Translating and the Computer 21, London: Aslib/IMI.

[38] Güvenir, H. A. and I. Cicekli. 1998. Learning Translation Templates from Examples. Information Systems 23, 353–363.

[39] Güvenir, H. A. And A. Tunç. 1996. Corpus-Based Learning of Generalized Parse Tree Rules for Translation. In G. McCalla (ed.) Advances in Artificial

Intelligence, 121–132, Berlin: Springer Verlag.

[40] Sato, S. And M. Nagao. 1990. Toward Memory-Based Translation. Coling (1990), Vol. 3, 247–252.

[41] Sumita, E.K. Oi, O.Furuse, H.Lida, T. Higuchi, N.Takahashi and H. Kitano. 1993.Example-Based Machine Translation on Massively Parallel Processors. IJCAI-93 Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, Chambéry, France, 1283–1288.

[42] Boitet, C. and Zaharin, Y. (1988). Representation trees and string-tree correspondences", In Proceedings. of COLING-88, Budapest

[43] Tang, E. K. And Zaharin, Y. (1995). "Handling Crossed Dependencies with the STCG". In Proceedings of NLPRS'95, Seoul.

[44 Mosleh Al-Adhaileh, Tang E.K Zaharin Yusoff. A Synchronization Structure of SSTC and Its Applications In Machine Translation

[45] http://www.cmpe.boun.edu.t

[46] http://www.cs.bilkent.edu.tr/

[47] http://www.ceng.metu.edu.tr/

[48] http://www.ce.itu.edu.tr/

[49] https://zemberek.dev.java.net

[50] http://nlp.stanford.edu/index.shtml

[51] Sumita, E. and H. Iida. 1991. Experiments and Prospects of Example-Based Machine Translation. 29th Annual Meeting of the Association for Computational Linguistics, Berkeley, California, 185–192.

[52] Sato, S. 1993, Example-Based Translation of Technical terms. TMI (1993), 58–68

[53] Murata, M., Q. Ma, K. Uchimoto and H. Isahara. 1999. An Example-Based Approach to Japanese-to-English Translation of Tense, Aspect, and Modality. TMI (1999), 66–76.

APPENDIX A

CORPUS	
Add margins for binding	Ciltleme için kenar boşlukları ekleyin.
Click a category in the Categories box	Kategoriler kutusunda, bir kategoriyi tıklatın.
Click a paper size	Bir sayfa boyutu tıklatın
Click Assign	Ata'yı tıklatın.
Click Close	Kapat'ı tıklatın.
Click Create New.	Yeni Oluştur'u tıklatın.
Click Footnotes or Endnotes.	Dipnotlar veya Sonnotlar'ı tıklatın.
Click Heading number to insert the chapter number, and then click Insert.	Bölüm numarası eklemek için Başlık numarası seçeneğini ve sonra Ekle'yi tıklatın.
Click Heading text to insert the chapter heading, and then click Insert.	Bölüm başlığı eklemek için Başlık metni seçeneğini ve sonra Ekle'yi tıklatın.
Click in the document where you want to insert the AutoText entry.	Belge içinde Otomatik Metin girişini eklemek istediğiniz yeri tıklatın.
Click in the section for which you want to create a different header or footer.	Farklı üstbilgi veya altbilgi oluşturmak istediğiniz bölümü tıklatın.
Click Insert, and then click Close	Ekle'yi tıklatın ve sonra Kapat'ı tıklatın.
Click Insert, point to Reference, and then click Footnote.	Ekle'yi tıklatın, Başvuru'yı işaret edin ve Dipnot'u tıklatın.
Click New.	Yeni'yi tıklatın.
Click OK.	Tamam'ı tıklatın.
Click Page break, and then click OK.	Sayfa sonu'nu tıklatın ve sonra Tamam'ı tıklatın.
Click Portrait or Landscape	Dikey'i veya Yatay'ı tıklatın.
Click Remove.	Kaldır'ı tıklatın.
Click Run.	Çalıştır'ı tıklatın.
Click the arrow next to the Zoom box.	Yakınlaştırma kutusunun yanındaki oku tıklatın
Click the Commands tab.	Komutlar sekmesini tıklatın.
Click the Layout tab.	Düzenleme sekmesini tıklatın.
Click the name of the AutoText entry you want.	İstediğiniz Otomatik Metin girdisinin adını tıklatın.
Click the Options tab.	Seçenekler sekmesini tıklatın.
Click the Toolbar Options arrow	Araç Çubuğu Seçenekleri okunu tıklatın.

Click the toolbar you want to display.	Görüntülemek istediğiniz araç çubuğunu seçin.
Click the Toolbars tab.	Araç Çubukları sekmesini tıklatın
Click to the left of the index, table of authorities, or table of figures that you want to delete.	Silmek istediğiniz dizin, kaynakça veya şekiller tablosunun soluna tıklatın.
Click to the left of the index, table of authorities, or table of figures that you want to update.	Güncelleştirmek istediğiniz dizinin, kaynakçanın veya şekiller tablosunun solunu tıklatın.
Click where you want the items to be pasted.	Öğenin yapıştırılmasını istediğiniz yeri tıklatın.
Click where you want the reference located.	Başvurunun yerleşmesini istediğiniz yeri tıklatın.
Click where you want to insert a section break.	Bölüm sonu eklemek istediğiniz yeri tıklatın.
Create the header or footer for the first page of the document or section.	Belgenin veya bölümün ilk sayfası için üstbilgi veya altbilgi oluşturun.
Footnotes and endnotes appear at the end of the Web page.	Dipnotlar ve sonnotlar Web sayfasının sonunda görünür.
For Help on an option, click the question mark , and then click the option.	Bir seçenek hakkında Yardım için, soru işaretini ve sonra seçeneği tıklatın.
Format the new reference as superscript	Yeni başvuruyu üst simge olarak biçimlendirme
Formatting inconsistencies will be marked with blue, wavy underlines.	Biçimlendirme tutarsızlıkları mavi dalgalı alt çizgiler olarak işaretlenecektir.
I can't see the note pane at the bottom of the page.	Sayfa sonunda not bölmesini göremiyorum.
In the Apply to box, click Selected text.	Uygula kutusundaki Seçili metin'i tıklatın.
In the box to the right, click the name of the command or other item.	Sağdaki kutudan komutun adını veya diğer öğeyi tıklatın.
In the browser, custom note separators appear as short horizontal lines.	Tarayıcıda, özel not ayırıcıları kısa yatay çizgiler olarak görünür.
In the Categories box, click a category for the command	Kategoriler kutusunda, komut için bir kategori seçin.

In the Categories box, click Built-in Menus.	Kategoriler kutusunda, Yerleşik Menüler'i tıklatın.
In the Categories box, click the category that contains the command or other item.	Kategoriler kutusundan komutu veya diğer öğeyi içeren kategoriyi tıklatın.
In the document, rest the pointer on the note reference mark.	Belgenin içinde, işaretçiyi not başvuru işaretinin üzerinde tutun.
In the For which box, click the note to which you want to refer.	Hangisi kutusundan başvurmak istediğiniz notu tıklatın.
In the Gutter box, enter a value for the gutter margin	Cilt Payı kutusuna kenar boşluğu için bir değer girin.
In the Gutter position box, click Left or Top.	Cilt payı yeri kutusu içinde, Sol veya Üst'ü tıklatın.
In the Insert reference to box, click Footnote number or Endnote number.	Başvurulacak ekle kutusundan Dipnot numarası seçeneğini veya Sonnot numarası seçeneğini tıklatın.
In the Insert reference to box, select what you want to insert in the header or footer.	Başvuru ekle kutusunda, üstbilgiye ve altbilgiye ne eklemek istediğinizi seçin.
In the Macro name box, click ListCommands.	Makro adı kutusunda Liste Komutları'nı tıklatın.
In the Macros in box, click Word commands.	Makro yeri kutusunda Word komutları'nı tıklatın.
In the Name box, enter the sound you want.	Ad kutusunda, istediğiniz sesi girin.
In the note pane, click All footnotes or All endnotes.	Not bölmesinde, Tüm dipnotlar veya Tüm sonnotlar seçeneğini tıklatın.
In the Number format box, click the option you want.	Rakam biçimi kutusundan istediğiniz seçeneği tıklatın.
In the Reference type box, click Footnote or Endnote.	Başvuru türü kutusundan Dipnot veya Sonnot'u tıklatın.
In the Sound Events box, click the event you want.	Ses Özellikleri kutusunda, istediğiniz özelliği tıklatın.
In the Style type box, click Character.	Stil türü kutusunda Karakter'i tıklatın.
In the Style type box, click List.	Stil türü kutusunda Liste'yi tıklatın.

In the Style type box, click Paragraph.	Stil türü kutusunda Paragraf'ı tıklatın.
In the Style type box, click Table.	Stil türü kutusunda Tablo'yu tıklatın.
In the Styles and Formatting list, click Footnote Reference or Endnote Reference.	Stiller ve Biçimlendirme listesinde, Dipnot Başvurusu'nu veya Sonnot Başvurusu'nu tıklatın.
In the Styles and Formatting task pane, click Custom in the Show box.	Stiller ve Biçimlendirme görev bölmesinde, Göster kutusundaki Özel'i tıklatın.
In the Toolbar name box, type the name you want, and then click OK	Araç çubuğu adı kutusunda, istediğiniz yeni bir adı yazın ve sonra Tamam'ı tıklatın.
In your document, right-click the blue, wavy underline where a formatting inconsistency has occurred.	Belgenizde biçimlendirme tutarsızlığının olduğu yeri gösteren mavi, dalgalı alt çizgiyi sağ tıklatın.
Make your changes to the header or footer.	Üstbilgi veya altbilgideki değişikliklerinizi uygulayın.
Many accessibility features are built right into Microsoft Word.	Birçok erişilebilirlik özelliği Microsoft Word'de hazır bulunmaktadır.
Many features and commands are available directly from the keyboard.	Birçok özellik ve komut doğrudan klavye aracılığıyla kullanılabilir.
Margins of the left page are a mirror image of those on the right page.	Sol sayfanın kenar boşluğu ayarları sağ sayfa ayarlarının yansımasıdır.
Microsoft Word offers several page margin options.	Microsoft Word'de çeşitli sayfa kenar boşluğu seçenekleri vardır.
My footnotes are getting renumbered in my table.	Dipnotlarım tablomda yeniden numaralandırılıyor.
Note text does not appear with the separator.	Not metni, ayırıcıyla birlikte görünmez.
On the Customize dialog box, click Close.	Özelleştir iletişim kutusunda, Kapat'ı tıklatın.
On the File menu, click New.	Dosya menüsünden Yeni'yi tıklatın.
On the File menu, click Print.	Dosya menüsünden Yazdır'ı tıklatın
On the File menu, click Print.	Dosya menüsünden Yazdır'ı tıklatın.
On the Format menu, click Reveal Formatting.	Biçim menüsünde, Biçimlendirmeyi Göster'i tıklatın.
On the Header and Footer toolbar, click Page Setup	Üstbilgi ve Altbilgi araç çubuğundaki Sayfa Yapısı'nı tıklatın.

On the Insert menu, click Break.	Ekle menüsünde Kes'i tıklatın.
On the Insert menu, point to Reference, and then click Cross-reference.	Ekle menüsündeki Başvuru'yu seçin ve Çapraz Başvuru'yu tıklatın.
On the Insert menu, point to Reference, and then click Footnote.	Ekle menüsündeki Başvuru'nun üzerine gelin ve Dipnot'u tıklatın.
On the Insert menu, point to Reference, click Index and Tables, and then click the tab you want.	Ekle menüsünden Başvuru'nun üzerine gelin, Dizin ve Tablolar'ı, sonra istediğiniz sekmeyi tıklatın.
On the Standard toolbar, click New Blank Document	Standart araç çubuğunda, Yeni Boş Belge'yi tıklatın.
On the Tools menu, click Customize.	Araçlar menüsünde, Özelleştir'i tıklatın.
On the Tools menu, click Options, and then click the View tab.	Araçlar menüsünde Seçenekler'i tıklatın ve sonra Görünüm sekmesini tıklatın.
On the View menu, click Footnotes.	Görünüm menüsünden Dipnotlar'ı tıklatın.
On the View menu, click Header and Footer.	Görünüm menüsünden Üstbilgi ve Altbilgi'yi tıklatın.
On the View menu, point to Toolbars.	Görünüm menüsünde, Araç Çubukları'nın üzerine gidin.
Page margins are the blank space around the edges of the page.	Kenar boşlukları, sayfa kenarlarındaki boş alanlardır.
Part of the footnote got moved to the next page.	Dipnotun bir bölümü bir sonraki sayfaya taşındı.
Place the insertion point in the section in which you want to change the footnote or endnote format.	Ekleme noktasını, biçimini değiştirmek istediğiniz sonnotun veya dipnotun olduğu bölüme yerleştirin.
Point to the left or right edge of the box	Kutunun sol veya sağ köşesinin üzerine gidin.
Select the Automatically update check box.	Otomatik olarak güncelleştir onay kutusunu seçin.
Select the first instance of formatting you want to compare.	Karşılaştırmak istediğiniz ilk biçimlendirme örneğini seçin.
Select the first item you want to copy.	Kopyalamak istediğiniz ilk öğeyi seçin.
Select the grammar and style rules used during a grammar check.	Dilbilgisi denetimi sırasında kullanılacak dilbilgisi ve stil kurallarını seçebilirsiniz.
Select the pages that you want to	Dikey veya yatay yönlendirmek istediğiniz

change to portrait or landscape orientation.	sayfaları seçin.
Select the separator and make changes.	Ayırıcıyı seçin ve değişiklikleri yapın.
Select the text whose formatting you want to review.	Biçimlendirmesini görmek istediğiniz metni seçin.
Select the text you want to change.	Değiştirmek istediğiniz metni seçin.
Size and zoom options	Boyut ve yakınlaştırma seçenekleri
Some of the footnotes or endnotes have disappeared.	Bazı dipnot veya sonnotlar yok oldu.
Superscript formatting is applied to the footnote.	Üst simge biçimlendirmesi dipnota uygulanır.
Switch to normal view if you are in a different view.	Farklı bir görünümdeyseniz, normal görünüme geçin.
Switch to normal view.	Normal görünüme geçin.
Switch to print layout view or Web layout view.	Yazdırma düzeni görünümüne veya Web düzeni görünümüne geçin.
The following procedure creates a new, blank document.	Aşağıdaki yordam yeni, boş bir belge oluşturur.
The formatting information will appear in the Reveal Formatting task pane.	Biçimlendirme bilgileri Biçimlendirmeyi Göster görev bölmesinde belirecektir.
The grammar checker may not look for all types of problems; it's designed to focus on those that are most typical or frequent.	Yazım denetleyicisinde olduğu gibi, içerik olarak farklı terimler bulunabilir.
The new default settings are saved in the template on which the document is based.	Yeni varsayılan ayarlar belgenin dayandığı şablon içine kaydedilir.
The note text appears above the mark in a ScreenTip.	Not metni bir Ekran İpucu'ndaki işaretin üst kısmında görünür.
The note text appears above the mark.	Not metni, işaretin üzerinde görünür.
The pointer shape indicates how the item will be formatted.	İşaretçinin şekli öğenin nasıl biçimlendirileceğini gösterir.
These features are available to everyone, without the need for additional accessibility aids.	Bu özellikler ek erişilebilirlik yardımı gerekmeksizin herkes tarafından kullanılabilir.

To add the new style definition to your template, select the Add to template check box.	Şablonunuza yeni stil tanımını eklemek için, Şablona ekle onay kutusunu işaretleyin.
To create a header, enter text or graphics in the header area.	Üstbilgi oluşturmak için üstbilgi alanına metin veya grafik girin.
To determine the minimum margin settings, check your printer manual.	Minimum kenar boşluğu ayarlarını belirlemek için, yazıcınızın el kitabını inceleyin.
To edit the separator, insert a Clip Art divider line or type text.	Ayırıcıyı düzenlemek için, Küçük Resim ayırıcı çizgisi ekleyin veya metin girin.
To remove the separator, press DELETE	Ayırıcıyı kaldırmak için, DELETE'e basın.
To restore the default separator, click Reset.	Varsayılan ayırıcıyı geri yüklemek için, Sıfırla'yı tıklatın.
To view more toolbars, click Customize.	Daha fazla araç çubuğu görüntülemek için, Özelleştir'i tıklatın.
Under Editing options, select or clear the Keep track of formatting check box.	Düzenleme seçenekleri altında, Biçimlendirmeyi izle onay kutusunu seçin veya temizleyin.
Under Editing options, select the Mark formatting inconsistencies check box.	Düzen seçeneklerinin altındaki Biçimlendirme tutarsızlıklarını işaretle onay kutusunu işaretleyin.
Under Formatting, select the options you want.	Biçimlendirme'de istediğiniz seçeneği belirleyin.
Under Margins, select the options you want.	Kenar Boşlukları altında, istediğiniz seçenekleri belirleyin.
Under Orientation, click Portrait or Landscape.	Yönlendirme'nin altındaki Dikey veya Yatay'ı tıklatın.
Under Page range, specify the portion of the document you want to print.	Sayfa aralığı altında belgenin yazdırmak istediğiniz kısmını belirtin.
Under Styles, click the style you want to change, and then click Modify.	Stil'den, değiştirmek istediğiniz stili ve sonra Değiştir'i tıklatın.
Use portrait and landscape orientation in the same document	Aynı belgede dikey ve yatay yönlendirme kullanma
View footnotes and endnotes in a	Microsoft Word belgesi içindeki dipnotları

Microsoft Word document	ve sonnotları görüntüleme
Word shares custom dictionaries with other Microsoft Office programs.	Word bu sözlükleri diğer Microsoft Office programlarıyla paylaşır.
You can also use keyboard shortcuts to zoom.	Yakınlaştırmak için klavye kısayollarını da kullanabilirsiniz.
You can customize color and sound options	Renk ve ses seçeneklerini özelleştirebilirsiniz.
You can customize separators by adding borders, text, or graphics.	Kenarlıklar, metin veya grafikler ekleyerek, ayırıcıları özelleştirebilirsiniz.
You can customize toolbars and menu commands	Araç çubuklarını ve menü komutlarını özelleştirebilirsiniz.
You can customize Word to better suit your needs	Word'ü ihtiyaçlarınıza daha iyi karşılık verecek şekilde özelleştirebilirsiniz
You can even create a custom toolbar button or menu command	Ayrıca özel araç çubuğu düğmeleri ve menü komutları da oluşturabilirsiniz.
You might use footnotes for detailed comments and endnotes for citation of sources.	Dipnotları detaylı açıklamalar, sonnotları ise kaynakları belirtmek için kullanabilirsiniz.
You must delete the note reference mark that corresponds to the note.	Nota ilişkin not başvuru işaretini silmeniz gerekmektedir.

APPENDIX B

```
public boolean isLeaf() { return false;}
public boolean isPhrasal() { return false;}
public boolean equals(Object object) { return false;}
public int compareTo(Object object) { return 0;}
public int indexOf(Tree tree) { return 0;}
public List getChildrenAsList() { return null;}
public void setChildren(Tree[] treeArray) { }
public void setChildren(List list) { }
public Label label() { return null;}
public void setLabel(Label label) { }
public Tree firstChild() { return null;}
public Tree lastChild() { return null;}
public StringBuffer toStringBuffer(StringBuffer stringBuffer) { return null;}
public String toString() { return null;}
public String nodeString() { return null;}
public void pennPrint(PrintWriter printWriter) { }
public void pennPrint(PrintStream printStream) { }
public void pennPrint() { }
public int depth() { return 0;}
public Sentence yield() { return null;}
public List yield(List list) { return null;}
public Collection labels() { return null;}
public List subTreeList() { return null;}
public Collection subTrees(Collection collection) { return null;}
public Tree deepCopy() { return null;}
public Tree deepCopy(TreeFactory treeFactory) { return null;}
public static Tree valueOf(String string) throws IOException { return null;}
public void insertDtr(Tree tree, int _int) { }
```