IMPROVING THE PRECISION OF EXAMPLE-BASED MACHINE TRANSLATION BY LEARNING FROM USER FEEDBACK

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Turhan Osman Daybelge

September, 2007

ABSTRACT

IMPROVING THE PRECISION OF EXAMPLE-BASED MACHINE TRANSLATION BY LEARNING FROM USER FEEDBACK

Turhan Osman Daybelge
M.S. in Computer Engineering
Supervisor: Assist. Prof. Dr. İlyas Çiçekli
September, 2007

Example-Based Machine Translation (EBMT) is a corpus based approach to Machine Translation (MT), that utilizes the translation by analogy concept. In our EBMT system, translation templates are extracted automatically from bilingual aligned corpora, by substituting the similarities and differences in pairs of translation examples with variables. As this process is done on the lexical-level forms of the translation examples, and words in natural language texts are often morphologically ambiguous, a need for morphological disambiguation arises. Therefore, we present here a rule-based morphological disambiguator for Turkish. In earlier versions of the discussed system, the translation results were solely ranked using confidence factors of the translation templates. In this study, however, we introduce an improved ranking mechanism that dynamically learns from user feedback. When a user, such as a professional human translator, submits his evaluation of the generated translation results, the system learns "context-dependent co-occurrence rules" from this feedback. The newly learned rules are later consulted, while ranking the results of the following translations. Through successive translation-evaluation cycles, we expect that the output of the ranking mechanism complies better with user expectations, listing the more preferred results in higher ranks. The evaluation of our ranking method, using the precision value at top 1, 3 and 5 results and the BLEU metric, is also presented.

Keywords: Example-Based Machine Translation, Learning from User Feedback, Morphological Disambiguation.

ÖZET

KULLANICI GERİ BİLDİRİMİNDEN ÖĞRENEREK ÖRNEK TABANLI MAKİNE ÇEVİRİSİ HASSASİYETİNİ İYİLEŞTİRMEK

Turhan Osman Daybelge
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. İlyas Çiçekli
Eylül, 2007

Örnek Tabanlı Makine Çevirisi (ÖTMÇ), analojiyle çeviri kavramını kullanan, derlem tabanlı bir Makine Çevirisi (MÇ) yaklaşımıdır. Bizim ÖTMÇ sistemimizde çeviri şablonları, çift dilli, hizalanmiş derlemlerden otomatik olarak, çeviri örneği çiftleri arasındaki benzerlik ve farklılıkları değişkenler ile değiştirerek elde edilir. Bu işlem esnasında çeviri örneklerinin morfolojik açıdan çözümlenmiş halleri kullanılır. Çoğu zaman, doğal dil metinlerinde kelimeler morfolojik açıdan belirsiz oldukları için, bu belirsizliği giderecek bir araca ihtiyaç duyulur. Bu yüzden, Türkçe için kural tabanlı bir morfolojik belirsizlik giderici geliştirdik. Tartışılan sistemin önceki sürümlerinde, çeviri sonuçları yalnızca çeviri şablonlarının güven çarpanları kullanılarak sıralanıyordu. Bu çalışmada kullanıcı geri bildiriminden öğrenen, geliştirilmiş bir sonuç sıralama mekanizmasını takdim ediyoruz. Bir kullanıcı, örneğin profesyönel bir çevirmen, çeviri sonuçları hakkındaki değerlendirmelerini girdiğinde, sistem bu geri bildirimden "bağlama bağlı birlikte kullanım kuralları" öğrenir. Bu kurallara, takip eden çeviri işlemlerinin sonuç sıralama aşamalarında başvurulur. Birbirini izleyen çeviri-değerlendirme döngülerinin sonucunda, sıralama mekanizması çıktısının, tercih edilen sonuçların üst sıralarda listelenmesi açısından, kullanıcı beklentilerini daha iyi karşılayan bir hale gelmesini bekliyoruz. Sıralama mekanizmasının, en üstteki 1, 3 ve 5 sonuç için hassasiyet değerlerini ve BLEU ölçüsünü kullanarak yapılmış değerlendirmesini sunuyoruz.

Anahtar sözcükler: Örnek Tabanlı Makine Çevirisi, Kullanıcı Geri Bildiriminden Öğrenmek, Morfolojik Belirsizlikleri Gidermek.

Acknowledgement

I am deeply indebted to my advisor Assist. Prof. Dr. İlyas Çiçekli for his support, guidance and assistance with this work. My thanks are due to Prof. Dr. Fazlı Can for kindly reviewing this work, and also for encouraging and supporting me throughout my studies. I would like to express my gratitude also to my former teacher Prof. Dr. Avadis Hacınlıyan for directing me to the field of natural language processing. I would like to thank Dr. Ayşenur Birtürk for accepting to read and review this thesis. I also acknowledge the Scientific and Technological Research Council of Turkey (TÜBİTAK) for supporting my studies under the M.S. Fellowship Program.

I finally wish to thank my family, for their continual support and motivation during my studies. Acknowledgement and thanks also to my girlfriend, Deniz, who has always been there for me.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Translation process between two natural languages consists of basically two stages. These are the interpretation of the meaning of a text in a source language, and the reproduction of an equivalent text that conveys the same message in a target language. The first stage is realized through a mapping of a given set of linguistic elements (words, phrases, syntax) of the source language into some semantic representations of objects, concepts and actions in the translator's mind, acquired from his real world experiences. Similarly, in the second stage, the translator maps those semantic representations back into some other linguistic elements, but this time to that of the target language. The critical problem here is that, generally, neither the mapping rules nor the semantic representations in the translator's mind are formally well-defined.

Since language and its translation are rather complex human phenomena, any serious study must at some point decompose them into a series of levels of abstractions. The linguistic strata usually considered in such abstractions have been: phonology, morphology, syntax, semantics and pragmatics, each dealing with a self-contained domain, and interacting with other levels in limited ways.

Translation task is indeed a challenging one even for an experienced translator. No word-for-word relationship exists between any two languages. Hence, mistranslations may easily happen when, for example, a word in the source language has multiple meanings, each of which represented with a distinct word in the target language. In such situations, in order to achieve an accurate translation, the translator first has to identify the correct concept referred by the ambiguous words, which is not necessarily a simple task. An obvious example is given in [14]:

> *The Latin translator of the Bible encountered the phrase which in Hebrew means "and rays glowed from Moses' face". Since in Hebrew "rays" and "horns" are referred to by the same word ("karnayim"), the translator selected the Latin word for "horns", and mistranslated the sentence as "and horns grew on Moses' head".* [...] *Such a failure, due to the confusion of concepts with words, resulted in the little horns on the head of Michelangelo's sculpture of Moses.*

Some of the linguists were led by similar examples and theoretical problems to the view that translation between natural languages is not even possible, as expressed in its most radical form by the Sapir-Whorf hypothesis. Sapir asserted in 1929 that "*The 'real world' is to a large extent unconsciously built up on the language habits of the group.* [...] *The worlds in which different societies live are distinct worlds, not merely the same world with different labels attached.*" [31].

What has become known as the Sapir-Whorf hypothesis is not generally applied in its strongest form, as it would imply, contrary to our observations in the real-world, the impossibility of meaningful communication between members of different societies. "*Nevertheless, it is considered that, this different perception and mental organisation of reality can be used to explain the existence of certain "gaps" between languages, which can turn translation into a very difficult process. Translators have to be aware of these gaps, in order to produce a satisfactory target text.*" [11]

Whereas, Chomsky's theory of Universal Grammar [7], explaining how children acquire their languages, claimes the existance of some universal principles for grammar rules that are common to all natural languages. Although, Chomsky did not attempt to apply his theory to translation, several other scholars built upon Chomsky's theory to support the universal translatability notion. Several of the well-known twentieth-century linguists including Jakobson, Bausch, Hauge, Nida and Ivir adopt the view that, essentially, everything can be expressed in any language, and therefore we can expect them to be mutually translatable [11]. Supporters of this view argue that the translatability of a text is guaranteed by the existence of universal syntactic and semantic categories. They further assert that [14]:

(i) Language is a means describing reality, and as such can and should expand to include newly discovered or innovated objects in reality.

(ii) Any word has a referent in reality, however indirectly. All concepts can be described by their manifestations in reality. For example, "empirical" means "based on observable phenomena." Even religious concepts, supposedly based on faith, can be described.

(iii) Translation is the transfer of conceptual knowledge from one language into another. It is the transfer of one set of symbols denoting concepts into another set of symbols denoting the same concepts. This process is possible because concepts have specific referents in reality. Even if a certain word and the concept it designates exist in one language but not in another, the referent this word and concept stand for nevertheless exists in reality, and can be referred to in translation by a descriptive phrase or neologism.

These optimistic or somewhat reductionist views, however, must be contrasted with those of some major philosophers of the 20th century, such as Wittgenstein, Quine, Heidegger and Gadamer, who were involved in the analysis and philosophy of language and, in particular, understanding. They have pointed out the complexity of the problem of interpretation of a text by the reader or a translator.

Hermeneutics, a branch of continental European philosophy with a long tradition concerned with human understanding and the interpretation of written texts, offers insights that may contribute to the understanding of meaning, translation, architectures for natural language understanding, and even to the methods suitable for scientific inquiry in Artificial Intelligence (AI) [22].

An earlier author of modern hermeneutics was Schleiermacher who taught from 1805 onwards at the universities of Halle and Berlin. Schleiermacher's concept of understanding holds empathy as well as intuitive linguistic analysis. He assumed that understanding is not merely the decoding of encoded information, but interpretation is built upon understanding, and it has a grammatical, as well as a psychological moment. Schleiermacher claimed that a successful interpreter could understand the author as well as, or even better than, the author understood himself because the interpretation reconstructs and explicates the hidden motives, implicit assumptions and strategies of the author [22].

Dilthey, who was initially influenced by Schleiermacher, began to emphasize that texts and actions were as much products of their times as expressions of individuals, and their meanings were consequently constrained by both an orientation to values of their period and a place in the web of their authors' plans and experiences. Thus he extended hermeneutics even further by relating interpretation to all historical objectifications. As such understanding moves from the outer manifestations of human action and productivity to explore their inner meaning. In his essay, *"The Understanding of Others and Their Manifestations of Life"* (1910) [12], Dilthey makes it clear that this move from outer to inner, from expression to what is expressed, is not based on empathy. Empathy is based on a direct identification with the other. Interpretation, on the other hand, involves an indirect or mediated understanding that can only be attained by placing human expressions in their historical context. Understanding is not a process of reconstructing the state of mind of the author, but one of articulating what is expressed in the work [21].

Martin Heidegger's *"Being and Time"* (1927) [16] completely transformed the discipline of hermeneutics. His philosophical hermeneutics shifted the focus from interpretation to existential understanding, which was treated more as a direct, non-mediated, thus in a sense more authentic way of being in the world than simply as a way of knowing. Advocates of this approach claim that such texts, and the people who produce them, cannot be studied using the same scientific methods as the natural sciences, thus use arguments similar to that of antipositivism. Moreover, they claim that such texts are conventionalized expressions of the experience of the author; thus, the interpretation of such texts will reveal something about the social context in which they were formed, but, more significantly, provide the reader with a means to share the experiences of the author. Among the key thinkers of this approach is the sociologist Max Weber [22].

According to Gadamer, words, that is, talk, conversation, dialogue, question and answer, produce worlds. In contrast to a traditional, Aristotelian view of language where spoken words represent mental images and written words are symbols for spoken words, Gadamerian perspective on linguistics emphasizes a fundamental unity between language and human existence. Interpretation can never be divorced from language or objectified. Because language comes to humans with meaning, interpretations and understandings of the world can never be prejudice-free. As human beings, one cannot step outside of language and look at language or the world from some objective standpoint. Language is not a tool which human beings manipulate to represent a meaningful world; rather, language forms human reality [4].

Modern ideas on hermeneutics hold that the writer may be an editor or a redactor and that he may have used sources. In considering this aspect of discourse one must take into account the writer's purpose in writing as well as his cultural milieu. Secondly, one must consider the narrator in the writing who can be different from the writer. Sometimes he is a real person, sometimes fictional. One must determine his purpose in speaking and his cultural milieu, taking into consideration the fact that he may be omnipresent and omniscient. One must also take into consideration the narratee within the story and how he hears. But even then one is not finished. One must reckon with the person or persons to whom the writing is addressed; the reader, not always the same as the one to whom the writing is addressed; and later readers. Thirdly, one must consider the setting of writing, the genre (whether poetry, narrative, prophecy, etc.), the figures of speech; the devices used, and, finally, the plot [15]. The coverage of the discipline of hermeneutics has since broadened to almost all texts, including multimedia and to understanding the bases of meaning.

Translation between natural languages has a long history, dating back to the earliest encounters of people from other countries like travelers, traders, artisans, politicians, or missionaries who spoke different languages, but wished, to communicate their messages, or to reach an understanding or an agreement with the foreigners. In our day due to the immensity of international relations the need for translation of various texts of literary, scientific, judicial, diplomatic etc. origin written or spoken in hundreds of languages has reached such a level, that its solution should be sought through extra-human means.

Indeed, the concept of Machine Translation (MT) emerged shortly after the end of the World War II, when the idea of automatic translation of texts between natural languages came into the minds of scientists such as Warren Weaver [34] and Alan Turing. Turing was among the ones, who deciphered the codes encrypted by the Enigma machines used in German naval communication. In this period, natural language was considered to be a code, and translation was analogous to code-breaking. Therefore, achieving automatic translation was seen as a matter of discovering some mechanical translation approach inspired by the modern cryptanalysis techniques developed at that time.

However today, machine translation systems are still far from replacing expert human translators, due to the complexities involved in the process of translation as discussed above. On the other hand, MT has proven to be successful in especially restricted domains such as the translation of weather reports or highly standardized texts such as legal documents. Also, when the goal is to get the grisp of a text, such as the content of a web page, and ungrammatical sentences are tolerable, MT constitutes a quick and inexpensive solution. With the future developments in the methods of AI and the computer technology, we may expect that machine translation will approach the level of expectations placed upon it.

Machine translation systems are generally categorized in two different aspects. The first categorization considers the architectural basis on which the MT systems are built upon. MT systems differ in the level to which they analyze their inputs, where the levels are often figurized as a pyramid diagram[1], such as the one in

[1]First appeared in [33].

Figure 1.1.

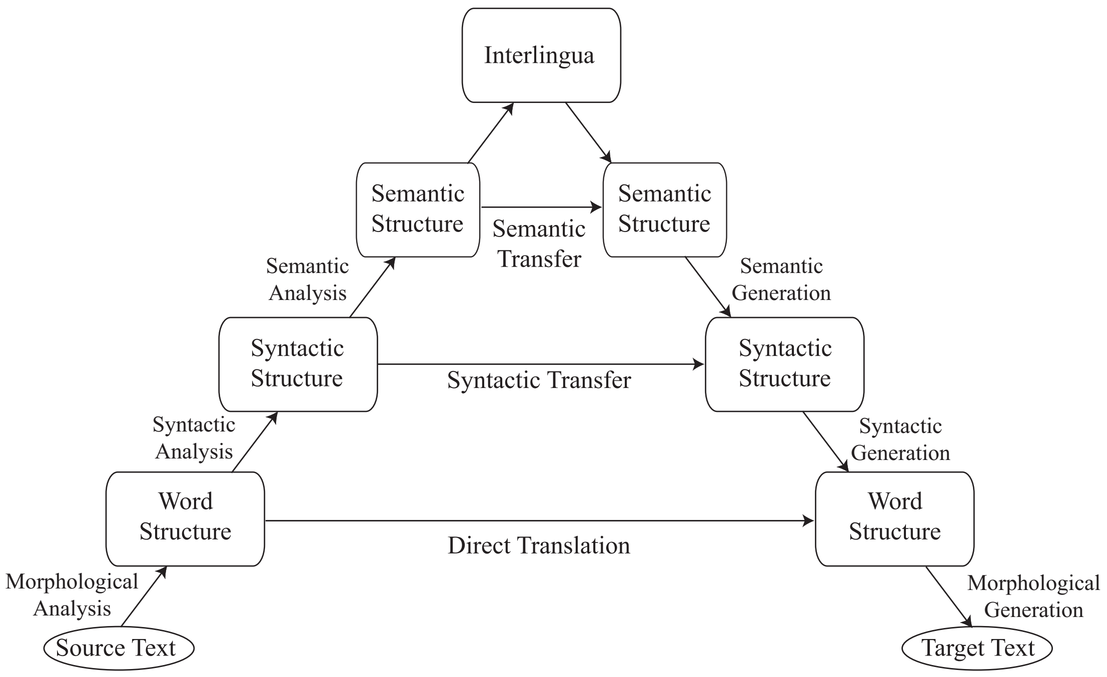


Figure 1.1: Vauquois' Pyramid.

The lowest level of the pyramid corresponds to *direct translation*, which uses only a dictionary and a few simple word-ordering rules, and translates a text solely by replacing each word in the source language with its most common translation in the target language. Direct translation performs minimal analysis on the input text, and thus is the simplest MT approach available. As expected, it has a limited success rate.

At the other extreme is the *interlingual translation* approach. In this approach, the input text is morphologically, syntactically and semantically analyzed and finally parsed into an interlingual representation that is independent both from the source and the target languages. Given the necessary generators of an arbitrary target language, the translation to that language can be achieved directly from the interlingual representation, without the need of language-pair dependent transfer rules. The drawback of the interlingual approach is the expense of complex analyses required. Especially the analyses depicted in the upper levels of the Vauquois' pyramid, such as the semantic analysis, requires real-world

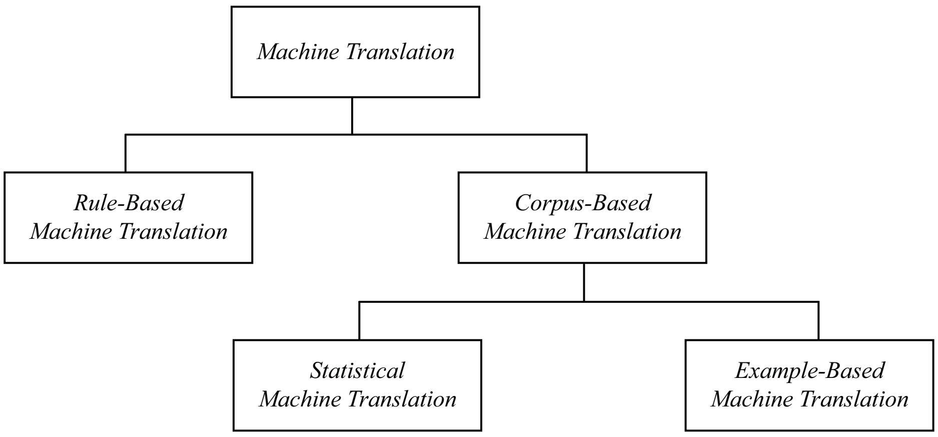

Machine Translation
Rule-Based Machine Translation
Corpus-Based Machine Translation
Statistical Machine Translation
Example-Based Machine Translation

Figure 1.2: Classification of the Machine Translation Systems.

knowledge, which has its own problems in terms of efficient acquisition, representation and storage. In spite of these difficulties, a semantic analysis, inspired by the philosophy of hermeneutics and supported by modern artificial intelligence techniques, would no doubt improve the quality of the translation.

Any approach in between the direct and interlingual translation options is included in the *transfer-based translation* category. In transfer-based approach, the source text is first parsed into an internal representation that is source language dependent, which is then converted into a corresponding internal representation specific to the target language. The tranfer rules are often language-pair dependent and motivated by linguistic concerns.

The second way of categorizing machine translation systems differentiates the approaches according to their means of acquiring the information used to translate the inputs. According to this scheme, there are two broad categories, namely, *rule-based* and *corpus-based* approaches, as depicted in Figure 1.2.

In the rule-based category, translation is done using hand-crafted rules that capture the grammatical correspondences between the languages. This approach requires a vast amount of translation rules, whose preparation is time consuming and requires expertise.

Corpus-based approaches, on the other hand, use a bilingual corpora to obtain the information required for translation. One of the corpus-based approaches is

Statistical Machine Translation [23]. The idea of applying the statistical and cryptanalytic techniques, then emerging in the field of communication theory, to the problem of machine translation was first proposed in 1949 by Warren Weaver in [34]. In statistical machine translation, translation results are generated on the basis of statistical models, whose parameters are derived from the analysis of bilingual corpora. For an input, the statistical translation models allow the system to generate many possible translations, among which the result with the highest probability is chosen.

Another corpus-based approach to MT is *Example-Based Machine Translation* (EBMT), which is regarded as an implementation of the case-based reasoning approach of machine learning. EBMT was first proposed by Nagao under the name *translation by analogy* [24]. Translation by analogy is a rejection of the idea that man translates sentences by applying deep linguistic analyses on them. Instead, it is argued, that man first decomposes the sentence into fragmental phrases, then translates these phrases into phrases in the target language, and finally composes these fragmental translations into a sentence. The translation of fragmental phrases is done in the light of prior knowledge, acquired in the form of translation examples.

In this thesis, we propose several improvements to an existing EBMT system [6, 13, 5]. We present here a new method for ranking the translation results generated by this system. Contrary to the previous versions, in our approach, the results ranking mechanism is dynamically trained by the user. User feedback is obtained in the form of an evaluation of the generated results. From the evaluation of the user, the system learns *context-dependent co-occurrence rules*, which are later consulted while ranking the results of the following translations. Through successive translation-evaluation cycles, we expect that the output of the ranking mechanism complies better with user expectations, listing the more preferred results in higher ranks.

1.1 Thesis Outline

The rest of this thesis is organized as follows: Chapter 2 provides a detailed review of the existing EBMT system. Chapter 3 describes several component of the system and the interactions among them. Chapter 4 presents a morphological disambiguator developed for Turkish, which is integrated into the translation system. Chapter 5 provides the details of the new results ranking mechanism. Chapter 6 discusses the results of the tests that are conducted to measure the effects of the newly added components. Chapter 7 concludes the thesis with a summary and a number of suggestions for further study.

Chapter 2

Review of the Current System

The system described in this thesis builds upon the recent papers of Çiçekli and Güvenir [6, 5], a detailed review of which is provided in this chapter. Using this system, translation can be done bidirectionally between two natural languages, such as Turkish and English. The translation system translates sentences from the source language to the target one using information gathered from previously observed translation examples.

The general structure of the system is given in Figure 2.1. The system has two main components which are learning and translation components. The learning component takes a bilingual corpus file as input and extracts translation templates which are to be used later by the translation component. When the learning is over, the templates extracted in the learning phase are stored in the file system. When a system user enters a phrase in one of the two languages, the translation component finds the most suitable translation templates for that phrase and performs the translation to the target language if possible. Each translation template is learned by the generalization of two translation examples. A simple example is given below:

$$\begin{aligned} \underline{\text{I am reading a}}\ \text{book} &\leftrightarrow \underline{\text{bir}}\ \text{kitap}\ \underline{\text{okuyorum}} \\ \underline{\text{I am reading a}}\ \text{newspaper} &\leftrightarrow \underline{\text{bir}}\ \text{gazete}\ \underline{\text{okuyorum}} \end{aligned} \tag{2.1}$$

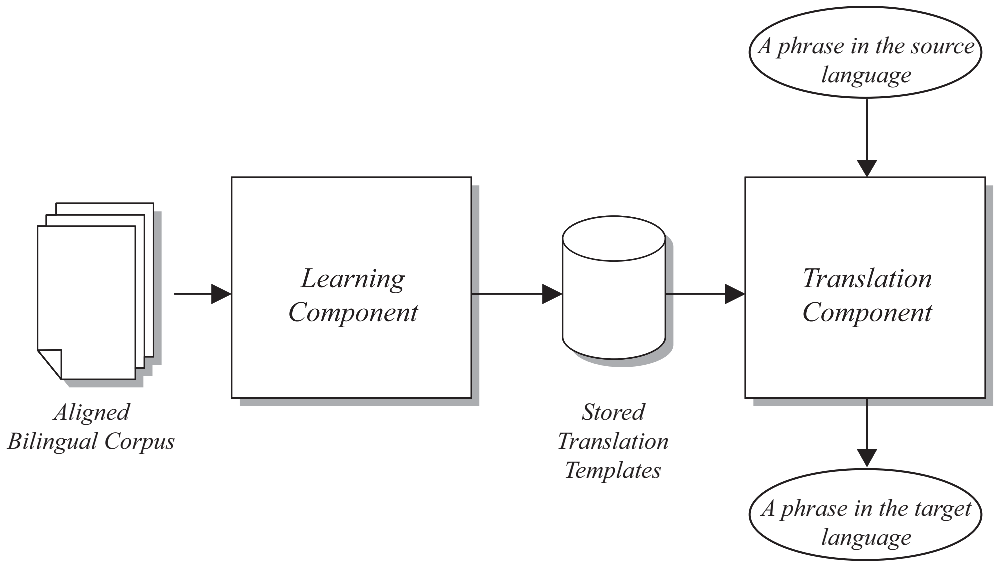

A phrase in the source language
Learning Component
Translation Component
Aligned Bilingual Corpus
Stored Translation Templates
A phrase in the target language

Figure 2.1: Basic Operation of the Translation System.

By analyzing the translation examples in (2.1), we can observe similarities (shown underlined) and differences on both sides. One of the heuristics, that is used to extract translation templates, is to replace differing parts by variables. Using this heuristic leads the system to learn the translation template shown in (2.2). A template, in which the differences are replaced by variables and similarities are kept untouched, such as the one below is called as *similarity translation template*.

$$\text{I am reading a } X \leftrightarrow \text{bir } Y \text{ okuyorum } \textbf{if } X \leftrightarrow Y \tag{2.2}$$

In addition to (2.2), we can also learn two more templates that represent the correspondence of the differing constituents of the examples as given below:

$$\begin{aligned} \text{book} &\leftrightarrow \text{kitap} \\ \text{newspaper} &\leftrightarrow \text{gazete} \end{aligned} \tag{2.3}$$

The templates that do not contain variables, such as those in (2.3), are named as *atomic translation templates* or shortly as *facts*.

2.1 Generating Match Sequences

We define a translation example as a pair of strings, $E^1 \leftrightarrow E^2$, where E^1 is in *language 1* and E^2 is in *language 2* and E^1 and E^2 are translations of each other. In order to be able to induce translation templates from two given translation examples $E^1_a \leftrightarrow E^2_a$ and $E^1_b \leftrightarrow E^2_b$, first a match sequence pair $M^1_{a,b} \leftrightarrow M^2_{a,b}$ (or shortly $M_{a,b}$) is generated where $M^1_{a,b}$ is a match sequence between E^1_a and E^1_b, and $M^2_{a,b}$ is a match sequence between E^2_a and E^2_b. A match sequence between two strings is defined as an alternating sequence of similarities and differences between those strings as depicted below:

$$S^1_0, D^1_0, S^1_1, \ldots, D^1_{n-1}, S^1_n \leftrightarrow S^2_0, D^2_0, S^2_1, \ldots, D^2_{m-1}, S^2_m, \text{ where } n = m > 1 \qquad (2.4)$$

A similarity, S^1_k, between two strings, E^1_a and E^1_b, is a non-empty sequence of tokens that are common to both strings. Similarly, a difference D^1_k between two strings is a token sequence pair ($D^1_{k,a}$, $D^1_{k,b}$) where $D^1_{k,a}$ is a substring of E^1_a and $D^1_{k,b}$ is a substring of E^1_b. Also no items in a similarity is allowed to appear in a difference. Any of S^1_0, S^1_n, S^2_0 or S^2_m can be empty, but S^1_i, for $0 < i < n$, and S^2_j, for $0 < j < m$, can not be empty. Furthermore, at least one similarity on both sides of $M_{a,b}$ must be non-empty. Under these restrictions, either a unique match sequence exists between the two strings, or no match sequences can be found [6].

As an example, the match sequence for the translation examples in (2.5) is given in (2.6).

$$\begin{aligned} \underline{\text{I came here}}\text{ today} \quad &\leftrightarrow \quad \underline{\text{buraya}}\text{ bugün }\underline{\text{geldim}} \qquad (2.5)\\ \underline{\text{I came here}}\text{ yesterday} \quad &\leftrightarrow \quad \underline{\text{buraya}}\text{ dün }\underline{\text{geldim}} \end{aligned}$$

$$\text{I came here (today, yesterday)} \leftrightarrow \text{ buraya (bugün, dün) geldim} \qquad (2.6)$$

The components of the match sequence (2.6) are given below:

S_0^1: I came here
D_0^1: (today, yesterday)
S_1^1: ϵ
S_0^2: buraya
D_0^2: (bugün, dün)
S_1^2: geldim

It can be seen that, in this example, $n = m = 1$ and the match sequence component S_1^1 is empty, as represented with ϵ.

Using the surface-level form representation of the translation examples may prevent us from extracting useful match sequences and degrade the generality of the translation templates learned. This problem becomes more critical when the source or target language is an agglutinative language such as Turkish which makes use of derivational and inflectional suffixes extensively. A typical example is given below:

$$\begin{array}{rcl} \underline{\text{I am}} \text{ coming} & \leftrightarrow & \text{geliyorum} \\ \underline{\text{I am}} \text{ going} & \leftrightarrow & \text{gidiyorum} \end{array} \tag{2.7}$$

From the translation examples of (2.7), we can not extract any match sequence since there are no similarities on the right hand sides in the surface-level form. To cope with this problem, we are keeping our translation examples in the lexical-level form which identifies morphemes such as root words and suffixes. Rewriting the examples given above in the lexical-level form yields 2.8. Here the *+PROG* morpheme represents the progressive tense suffix and the *+1SG* morpheme represents the 1st person singular agreement suffix.

$$\begin{array}{rcl} \underline{\text{I am}} \text{ come } \underline{\text{+PROG}} & \leftrightarrow & \text{gel } \underline{\text{+PROG +1SG}} \\ \underline{\text{I am}} \text{ go } \underline{\text{+PROG}} & \leftrightarrow & \text{git } \underline{\text{+PROG +1SG}} \end{array} \tag{2.8}$$

From the examples written in the lexical-level form, we can now extract the match sequence (2.9), that conforms with the previously stated restrictions.

$$\text{I am (come, go) +PROG} \leftrightarrow \text{ (gel, git) +PROG +1SG} \tag{2.9}$$

2.2 Learning Similarity Translation Templates

After extracting a match sequence from two given translation examples, the learning component tries to learn translation templates. Similarity translation templates are extracted by replacing the differences in the match sequence with variables. If there is only a single difference, D_0^1, on the left hand side and there is a single difference, D_0^2, on the right hand side of the match sequence, then the constituents of those differences should be the translations of each other. That is, $D_{0,a}^1 \leftrightarrow D_{0,a}^2$ and $D_{0,b}^1 \leftrightarrow D_{0,b}^2$. For example, since the match sequence (2.9) is in this form, the learning algorithm can derive the templates below from this match sequence.

$$\begin{aligned} \text{I am } X^1\text{+PROG} &\leftrightarrow Y^1 \text{ +PROG +1SG} \\ \text{come} &\leftrightarrow \text{gel} \\ \text{go} &\leftrightarrow \text{git} \end{aligned} \tag{2.10}$$

If there are $n > 1$ differences on each side, then in order to be able to extract a similarity translation template, we should be able to identify at least $n - 1$ correspondences between the differences in the left and right hand sides of the match sequence. If we can do that, the constituents of the remaining difference on the left hand side should be the translations of the constituents of the remaining difference on the right hand side.

After identifying the correspondences between the differences on the left and right hand sides, each pair of difference is replaced with a pair of variables. Algorithm 1 formalizes the process of similarity translation template learning.

Since the match sequence (2.9) has a single difference on both sides, the

```
SimilarityTTL($M_{a,b}$)
    • Assume that the match sequence $M_{a,b}$ for the pair of translation examples
    $E_a$ and $E_b$ is:
      $S^1_0, D^1_0, S^1_1, \ldots, D^1_{n-1}, S^1_n \leftrightarrow S^2_0, D^2_0, S^2_1, \ldots, D^2_{m-1}, S^2_m$
    if $n = m = 1$ then
      • Infer the following templates:
        $S^1_0 X^1 S^1_1 \leftrightarrow S^2_0 Y^1 S^2_1$ if $X^1 \leftrightarrow Y^1$
        $D^1_{0,a} \leftrightarrow D^2_{0,a}$
        $D^1_{0,b} \leftrightarrow D^2_{0,b}$
    else if $n = m > 1$ and $n - 1$ correspondences between differences in $M_{a,b}$
    are already known then
      • Assume that the unchecked corresponding difference pair is
        $(D^1_{k_n}, D^2_{l_n}) = ((D^1_{k_n,a}, D^1_{k_n,b}), (D^2_{l_n,a}, D^2_{l_n,b}))$.
      • Assume that the list of corresponding differences is
        $(D^1_{k_1}, D^2_{l_1}) \ldots (D^1_{k_n}, D^2_{l_n})$ including the unchecked ones.
      • For each corresponding difference $(D^1_{k_i}, D^2_{l_i})$,
        replace $D^1_{k_i}$ with $X^i$ and $D^2_{l_i}$ with $Y^i$ to get $M_{a,b}WDV$.
      • Infer the following templates:
        $M_{a,b}WDV$ if $X^1 \leftrightarrow Y^1$ and ... and $X^n \leftrightarrow Y^n$
        $D^1_{k_n,a} \leftrightarrow D^2_{l_n,a}$
        $D^1_{k_n,b} \leftrightarrow D^2_{l_n,b}$
    end if
```

Algorithm 1: SimilarityTTL. Extracts similarity translation templates.

learning algorithm can derive the templates in (2.10) from this match sequence without needing any prior knowledge.

On the other hand, for the match sequence in (2.12), which is extracted from the translation examples in (2.11) and has two differences on both hand sides, it is not possible to learn any translation templates without knowing the correspondence between the differences.

$$\begin{aligned} &\text{I } \underline{\text{drink +PAST}} \text{ tea} \leftrightarrow \text{çay } \underline{\text{iç +PAST}} \text{ +1SG} \\ &\text{you } \underline{\text{drink +PAST}} \text{ orange juice} \leftrightarrow \text{portakal suyu } \underline{\text{iç +PAST}} \text{ +2SG} \end{aligned} \tag{2.11}$$

$$\begin{aligned} &\text{(I, you) drink +PAST (tea, orange juice)} \leftrightarrow \\ &\text{(çay, portakal suyu) iç +PAST (+1SG, +2SG)} \end{aligned} \tag{2.12}$$

In order to be able to learn any translation templates, at least one of the correspondence pairs below should be known beforehand.

$$\begin{aligned} \text{I} \leftrightarrow \text{+1SG} \quad &, \quad \text{you} \leftrightarrow \text{+2SG} \\ \text{I} \leftrightarrow \text{çay} \quad &, \quad \text{you} \leftrightarrow \text{portakal suyu} \\ \text{tea} \leftrightarrow \text{çay} \quad &, \quad \text{orange juice} \leftrightarrow \text{portakal suyu} \\ \text{tea} \leftrightarrow \text{+1SG} \quad &, \quad \text{orange juice} \leftrightarrow \text{+2SG} \end{aligned} \tag{2.13}$$

Assuming that the correspondences “I ↔ +1SG” and “You ↔ +2SG” are known a priori, the similarity translation template learning algorithm extracts the templates given in (2.14). One should note that the corresponding variables, namely (X^1, Y^1), and (X^2, Y^2), are marked with identical superscripts.

$$\begin{aligned} X^1 \text{ drink +PAST } X^2 \quad &\leftrightarrow \quad Y^2 \text{ iç +PAST } Y^1 \\ \text{tea} \quad &\leftrightarrow \quad \text{çay} \\ \text{orange juice} \quad &\leftrightarrow \quad \text{portakal suyu} \end{aligned} \tag{2.14}$$

Some match sequences may have unequal number of differences on the left and right hand sides. Algorithm 1 can not learn any templates from this kind of match sequences. An example to this kind of match sequences is

$$\text{(I come, you go) +PAST} \leftrightarrow \text{ (gel, git) +PAST (+1SG,+2SG)} \tag{2.15}$$

This kind of match sequences that contain unequal number of differences on left and right hand sides occur frequently because of the different syntaxes of Turkish and English languages. To overcome this problem, the learning component should feed the similarity translation template learning algorithm with all possible instances of the match sequence with equal number of differences on left and right hand sides. In other words, the number of differences on the side that has fewer differences is increased by splitting at least one difference into two or more. For example, there is only one possible way of equalizing the number of differences of the match sequence (2.15), as shown below:

$$\text{(I, you)(come, go) +PAST} \leftrightarrow \text{ (gel, git) +PAST (+1SG,+2SG)} \tag{2.16}$$

For more complex examples, Algorithm 1 may fail to learn any translation template even if the number of differences on left and right hand sides of the match sequence are equal. In that case, the learning component incrementally increases the number of differences in the match sequence by one and tries to infer new translation templates. This process continues until a template is learned or no possible way of increasing the number of differences remains.

2.3 Learning Difference Translation Templates

Difference translation templates are the second kind of templates extracted by the learning component. While similarity translation templates replace differences in the match sequence with variables, difference translation templates do the opposite by substituting similarities. If there is a single similarity on both sides of the match sequence, then that pair of similarities should be the translations of each other. An example to this situation can be given as

$$\begin{gathered}\text{(I, you) drink +PAST (tea, orange juice)} \leftrightarrow \\ \text{(çay, portakal suyu) iç +PAST (+1SG, +2SG)}\end{gathered} \tag{2.17}$$

In this situation, the difference translation template learning algorithm replaces the similarities with variables. This form of the match sequence $M_{a,b}$, with similarities sustituted with variables is named as $M_{a,b}WDV$. By splitting the differences in $M_{a,b}WSV$ into two, the learning algorithm extracts two difference translation templates, namely M_aWSV: $(M_a^1WSV \leftrightarrow M_a^2WSV)$ and

```
DIFFERENCETTL($M_{a,b}$)
  if $numOfSim(M^1_{a,b}) = numOfSim(M^2_{a,b}) = n >= 1$ and
  $n-1$ corresponding similarities can be found in $M_{a,b}$ then
   • Assume that the unchecked corresponding similarity pair is
       ($S^1_{k_n}$,$S^2_{l_n}$).
   • Assume that the list of corresponding similarities is
       ($S^1_{k_1}$,$S^2_{l_1}$),. . . , ($S^1_{k_n}$,$S^2_{l_n}$) including the unchecked ones.
   • For each corresponding difference ($S^1_{k_i}$,$S^2_{l_i}$),
       replace $S^1_{k_i}$ with $X^i$ and $S^2_{l_i}$ with $Y^i$ to get $M_{a,b}WSV$.
   • Split $M_{a,b}WSV$ into $M_aWSV$ and $M_bWSV$ by seperating the differences.
   • Infer the following templates:
       $M_aWSV$ if $X^1 \leftrightarrow Y^1$ and . . . and $X^n \leftrightarrow Y^n$
       $M_bWSV$ if $X^1 \leftrightarrow Y^1$ and . . . and $X^n \leftrightarrow Y^n$
       $S^1_{k_n} \leftrightarrow S^2_{l_n}$
  end if
```
Algorithm 2: DIFFERENCETTL. Extracts difference translation templates.

M_bWSV: ($M^1_bWSV \leftrightarrow M^2_bWSV$). The difference translation templates extracted from (2.17) are

$$\begin{aligned} \text{I } X^1 \text{ tea} &\leftrightarrow \text{çay } Y^1 \text{ +1SG} \\ \text{you } X^1 \text{ orange juice} &\leftrightarrow \text{portakal suyu } Y^1 \text{ +2SG} \end{aligned} \tag{2.18}$$

In addition to the translation templates given above, the algorithm also learns the following atomic template.

$$\text{drink +PAST} \leftrightarrow \text{iç +PAST} \tag{2.19}$$

If there are $n > 1$ similarities on both sides of the match sequence, the difference translation template learning algorithm has to find the correspondence of at least $n-1$ similarities on the left and right hand sides of the match sequence in order to be able to infer any template. Algorithm 2 formalizes the process of difference translation template learning.

Some match sequences may have unequal number of similarities on the left and right hand sides. Algorithm 2 can not learn any template from this kind of match sequences, which occur frequently because of the different syntaxes of Turkish and

English languages. To overcome this problem, the learning component feeds the difference translation template learning algorithm with all possible instances of the match sequence with equal number of similarities on left and right hand sides. In other words, the number of similarities on the side that has fewer similarities is increased by splitting at least one similarity into two or more.

Still the learning algorithm may not infer any translation template even if the number of similarities on both sides of the match sequence are equal to each other. An example to this situation arises when the match sequence (2.21) is extracted from the following translation examples.

$$\begin{aligned} \underline{\text{I}}\text{ see }\underline{\text{+PAST the}}\text{ house} &\leftrightarrow \text{ev }\underline{\text{+ACC}}\text{ gör }\underline{\text{+PAST +1SG}} \\ \underline{\text{I}}\text{ break }\underline{\text{+PAST the}}\text{ mirror} &\leftrightarrow \text{ayna }\underline{\text{+ACC}}\text{ kır }\underline{\text{+PAST +1SG}} \end{aligned} \tag{2.20}$$

$$\begin{gathered} \text{I (see, break) +PAST the (house, mirror)} \leftrightarrow \\ \text{(ev, ayna) +ACC (gör, kır) +PAST +1SG} \end{gathered} \tag{2.21}$$

For the match sequence (2.21), no correspondence between the similarities on the left and right hand sides is valid. In such situations, the difference template learning algorithm incrementally increases the number of differences by one, until a template can be inferred, or there remains no possibility to divide a similarity. For the match sequence 2.20, there exists a single possibility for increasing the number of differences. By dividing the similarity "+PAST the" into "+PAST" and "the", and the similarity "+PAST +1SG" into "+PAST" and "+1SG", the learning algorithm can create a new instance of the match sequence with 3 similarities on both sides. Assuming that the correspondences I ↔ +1SG and +PAST ↔ +PAST are known, the learning algorithm can learn the following templates:

$$\begin{aligned} X^1\text{ see }X^2X^3\text{ house} &\leftrightarrow \text{ev }Y^3\text{ gör }Y^2Y^1 \\ X^1\text{ break }X^2X^3\text{ mirror} &\leftrightarrow \text{ayna }Y^3\text{ kır }Y^2Y^1 \\ \text{the} &\leftrightarrow \text{+ACC} \end{aligned} \tag{2.22}$$

2.4 Type Associated Template Learning

Although learning by substituting similarities or differences with variables yields templates that can be successfully used by the translation component, the templates are usually over generalized [5]. When the algorithm replaces some parts of the examples with variables, the type information of the replaced parts are lost. When used in translation, such a template may yield unwanted results, since the variables can represent any word or phrase. In order to overcome this problem, each variable is associated with a type information. An examplary template, the same one in (2.14), but this time marked with type information is given as

$$X^1_{Pron} \text{ drink +PAST } X^2_{Noun} \leftrightarrow Y^2_{Noun} \text{ iç +PAST } Y^1_{VERB-AGREEMENT} \tag{2.23}$$

In this example, the variable X^1_{Pron} can only be replaced by a pronoun and $Y^1_{VERB-AGREEMENT}$ can only be replaced by a verb agreement suffix.

In order to assign a type label to each variable, we have to have a mechanism that can decompose each word into its morphemes and identify root word and suffix categories. For this purpose we are using Turkish and English morphological analyzers in our translation system.

2.4.1 Learning Type Associated Similarity Templates

In order to assign a type label to a variable that substitutes a difference D_i, the learning component must inspect the constituents of this difference, namely $D_{i,a}$ and $D_{i,b}$. In general, the type of a root word is its part-of-speech category. For example, the type label of "book+Noun" would be simply "Noun". On the other hand, the type label of any morpheme that is not a root word would be its own name. For example, the type label of "+A1sg", which is the first person noun agreement morpheme in Turkish, is merely its own name, that is "A1sg". Assume that the learning algorithm tries to replace the difference D_i in 2.24 with a variable.

$$D_i\text{: (come+Verb, go+Verb)} \tag{2.24}$$

Observing that there is a single token in both of the constituents $D_{i,a}$ and $D_{i,b}$ and the types of the tokens match, the variable with type label would be X_{Verb}.

Although in some cases all of the type labels of tokens in $D_{i,a}$ and $D_{i,b}$ match, most of the times the situation will be different. Assume that this time the learning algorithm aims to replace the difference D_i below with a variable.

$$D_i\text{: (book+Noun +Sg, house+Noun +Pl)} \tag{2.25}$$

In this case, although the first pair of tokens of $D_{i,a}$ and $D_{i,b}$ match in terms of type, the second pair of tokens, "+Sg" and "+Pl", which are the singular and plural markers, do not match. In this kind of situations, the learning algorithm should be able to identify the supertype of "+Sg" and "+Pl". Given that the supertype of "+Sg" and "+Pl" is NOUN-SUF-COUNT, the variable that replaces the difference in (2.25) would be $X_{Noun\ NOUN-SUF-COUNT}$.

The hierarchical structure that represents the subtype-supertype relations between the type labels is modelled as a lattice in our system. There are two such lattices, one for *language 1* and the other for *language 2*. A section of the Turkish lattice used in the system is given in Fig 2.2. One should note that the lattice can be regarded as a directed acyclic graph (DAG), if each connection from a subtype to a supertype is considered to be a one directional arrow.

In the lattice there is a single node at the top of the hierarchy labelled "ANY". The leaf nodes are tokens that appear in the lexical-level form of the translation examples. Use of the lattice instead of a tree allows situations where a node has multiple parents such as the case of "+A3sg" which can both appear as the singular noun agreement and the 3^{rd} person singular verb agreement.

Using the lattice structure, the learning algorithm can assign a label to token

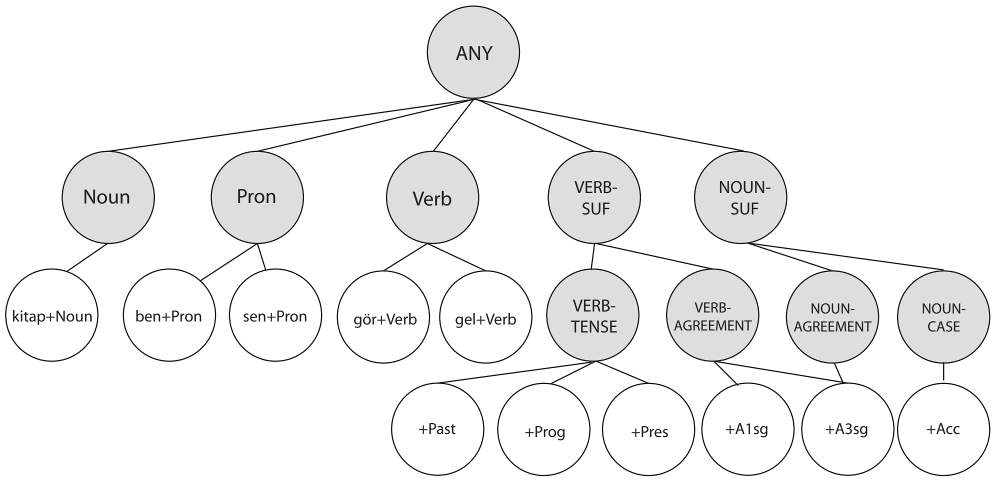


Figure 2.2: A Section of the Turkish Type Lattice.

pairs by finding the nearest common parent of the two tokens. Then the type label of a variable becomes the concatenation of each such token pair in the difference. An example of a difference, D_i can be given as

$$D_i\text{: (kitap+Noun +A3sg, ben+Pron +A1sg)} \tag{2.26}$$

Here, the type label of the first token pair, (kitap+Noun, ben+Pron), is "ANY" which is the nearest common parent of the two tokens. Likewise, the type label of the second token pair (+A3sg, +A1sg) will be VERB-AGREEMENT. So the label of the variable that replaces the difference D_i would be "ANY VERB-AGREEMENT".

2.4.2 Epsilon (ϵ) Insertion

In order to infer the type information of a variable in a similarity translation template, the learning algorithm looks into the lattice to find the nearest common parent of each token pair in the constituents of the associated difference. The type association algorithm defined above will fail when the constituents $D_{i,a}$ and $D_{i,b}$ of a difference D_i contain unequal number of tokens.

In cases where the constituents of a diffence contain unequal number of tokens, we can insert ϵ (empty string) tokens into the constituent with fewer tokens until the number of tokens are equalized. We can determine the insertion point of an epsilon token by calculating a generalization score for each of the possible insertion points and then choosing the one with the least score.

The generalization score of an epsilon insertion point possibility is calculated as the sum of the distances between the types of the corresponding tokens in the constituents of the difference after the epsilon insertion. The distances between token types are calculated using the lattice structures as the length of the shortest path between the types. The distance from epsilon to any type is set to 2.

Assume that the learning algorithm is going to assign a type label to the variable that is going to replace the difference in the following match sequence:

(a+Det +Indef +Sg red+Adj, the+Det +Def +SP blue+Adj) book+Noun +Sg (2.27)
↔ (bir+Num+Card kırmızı+Adj, mavi+Adj) kitap+Noun +A3sg +Pnon +Nom

In the difference on the left-hand side, there are 4 tokens in both of the constituents, hence there is no need for epsilon insertion. But, in the difference on the right-hand side, there are 2 tokens in the first constituent where there is a single token in the second one. In this case there are two epsilon insertion possibilities, i.e.,

(bir+Num+Card kırmızı+Adj, ϵ mavi+Adj) (2.28)
(bir+Num+Card kırmızı+Adj, mavi+Adj ϵ)

The section of the lattice that is going to be used to find the best position of epsilon insertion is given in Figure 2.3. The generalization scores for the two

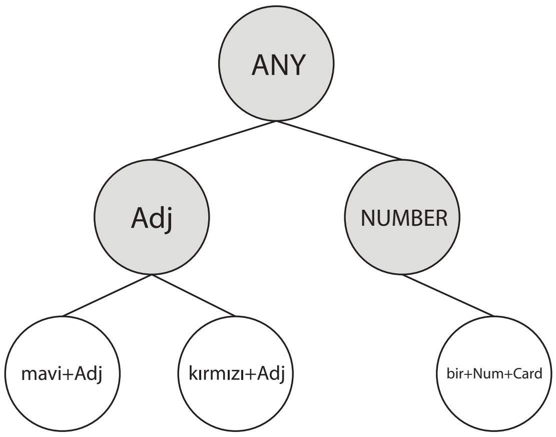


Figure 2.3: A Section of the Turkish Type Lattice.

epsilon insertion points are calculated as

$$\begin{aligned} genScore_1 &= minDist(\text{bir+Num+Card}, \epsilon) + minDist(\text{kırmızı+Adj, mavi+Adj}) \\ &= 2 + 2 = 4 \\ genScore_2 &= minDist(\text{bir+Num+Card, mavi+Adj}) + minDist(\text{kırmızı+Adj}, \epsilon) \\ &= 4 + 2 = 6 \end{aligned}$$

After the calculation, the epsilon insertion point with the smallest generalization score, in our case the first one, is chosen. Then the type of the variable will become "nullor(Num-Card) Adj". Here, the *nullor* function marks the token position as nullable, that is the token position can either be substituted with epsilon or a cardinal number during the translation phase. Given that the parent of "Def" and "Indef" is "DET-SUF" and the parent of "Sg" and "SP" is "DET-SUF-COUNT" in the English lattice, the similarity translation template and the two atomic templates that are learned from 2.27 then becomes

$$\begin{gathered} X^1_{Det\ DET-SUF\ DET-SUF-COUNT\ Adj}\ \text{book+Noun +Sg} \leftrightarrow \\ Y^1_{nullor(Num-Card)\ Adj}\ \text{kitap+Noun +A3sg +Pnon +Nom} \end{gathered} \tag{2.29}$$

$$\begin{aligned} \text{a+Det +Indef +Sg red+Adj} &\leftrightarrow \text{bir+Num+Card kırmızı+Adj} \\ \text{the+Det +Def +SP blue+Adj} &\leftrightarrow \text{mavi+Adj} \end{aligned}$$

2.4.3 Extension to the Previous Version: Learning Type Associated Difference Templates

The variable type labels for the similarity translation templates were inferred by generalizing the types of token pairs in the corresponding constituents of a difference. When it comes to learning type associated difference templates, one replaces similarities, which contain only a single constituent, with variables. In the previous versions of the translation system [5, 13], type associated difference template learning mechanism was not implemented, as generalizing type labels from a single constituent was not desired.

Abandoning the difference translation template learning feature would prevent us from learning useful information. Instead, we can choose to include this feature, but prevent the over-generalization of the type labels.

In type associated difference translation template learning, if the token is a root word, then the type of that token is determined as its parent in the type lattice[1]. On the other hand, if it is any other token that is not a root word, such as a feature structure property, then the type label of that token remains as its own name. In this way, the type labels are always determined strictly when compared to that of the similarity templates. For example, consider that we are trying to infer the type label of a variable that is going to be replaced by a similarity S_i: (kitap+Noun +A3sg +Pnon +Nom). Then the variable with its associated type label would be $X_{Noun\ A3sg\ Pnon\ Nom}$. This variable now represents any noun that is singular, without any possesive suffix and is in nominative case.

A type associated difference template learning example is given below. The

[1]Typically, in the type lattice, the parent of a root word will be its part-of-speech category.

match sequence for the following translation examples is given in (2.31).

$$\begin{aligned}
&\text{red+Adj } \underline{\text{book+Noun +Sg}} \leftrightarrow \\
&\qquad \text{kırmızı+Adj } \underline{\text{kitap+Noun +A3sg +Pnon +Nom}} \\
&\text{blue+Adj } \underline{\text{book+Noun +Sg}} \leftrightarrow \\
&\qquad \text{mavi+Adj } \underline{\text{kitap+Noun +A3sg +Pnon +Nom}}
\end{aligned} \tag{2.30}$$

$$\begin{aligned}
&\text{(red+Adj, blue+Adj) book+Noun +Sg} \leftrightarrow \\
&\qquad \text{(kırmızı+Adj, mavi+Adj) kitap+Noun +A3sg +Pnon +Nom}
\end{aligned} \tag{2.31}$$

Since there is a single similarity on both sides of the match sequence, the learning algorithm can replace them by variables without needing any prior knowledge. The templates learned from (2.31) are given below:

$$\begin{aligned}
\text{red+Adj } X^1_{Noun\ Sg} &\leftrightarrow \text{kırmızı+Adj } Y^1_{Noun\ A3sg\ Pnon\ Nom} \\
\text{blue+Adj } X^1_{Noun\ Sg} &\leftrightarrow \text{mavi+Adj } Y^1_{Noun\ A3sg\ Pnon\ Nom} \\
\text{book+Noun +Sg} &\leftrightarrow \text{kitap+Noun +A3sg +Pnon +Nom}
\end{aligned} \tag{2.32}$$

The above mentioned approach for associating types with variables has a flaw that has to be considered. For some match sequence instances, a learned difference template may be equivalent to the original translation example, that is used in learning that template. For example, from the translation examples in (2.33), we can extract the match sequence (2.35).

$$\begin{aligned}
&\text{red+Adj book+Noun } \underline{\text{+Sg}} \leftrightarrow \\
&\qquad \text{kırmızı+Adj kitap+Noun } \underline{\text{+A3sg +Pnon +Nom}}
\end{aligned} \tag{2.33}$$

$$\begin{aligned}
&\text{blue+Adj pencil+Noun } \underline{\text{+Sg}} \leftrightarrow \\
&\qquad \text{mavi+Adj kalem+Noun } \underline{\text{+A3sg +Pnon +Nom}}
\end{aligned} \tag{2.34}$$

$$\text{(red+Adj book+Noun, blue+Adj pencil+Noun) +Sg} \leftrightarrow \tag{2.35}$$
$$\text{(kırmızı+Adj kitap+Noun, mavi+Adj kalem+Noun) +A3sg +Pnon +Nom}$$

This will lead us to extract the following type associated difference templates:

$$\text{red+Adj book+Noun } X^1_{Sg} \leftrightarrow \text{kırmızı+Adj kitap+Noun } Y^1_{A3sg\ Pnon\ Nom} \tag{2.36}$$
$$\text{blue+Adj pencil+Noun } X^1_{Sg} \leftrightarrow \text{mavi+Adj kalem+Noun } Y^1_{A3sg\ Pnon\ Nom} \tag{2.37}$$
$$\text{+Sg} \leftrightarrow \text{+A3sg +Pnon +Nom} \tag{2.38}$$

While learning (2.38) will probably be useful, translation templates (2.36) and (2.37) are totally useless. Template (2.36) can **only** match the translation example (2.33), as it is equivalent in generality to the latter. The same is also true for the template (2.37), as it is equivalent to the translation example (2.34). So, there are no practical reasons for learning template (2.36) and (2.37). Therefore in our system, we prevent learning of such templates that do no generalization over the translation examples that are used to extract them. Thus, the only template that is going to be learned from this match sequence will be (2.38).

2.4.4 Learning from Previously Learned Templates

Although, extracting translation templates from translation example pairs, as it is presented in the previous sections, provide an effective learning method, the generality of the templates learned are usually limited. In order to increase the learning effectiveness, we do not only learn from example pairs, but also use the pairs of previously learned templates.

For example, assume that the translation templates in (2.39) have been learned from some translation examples. The first thing to do is to extract a match sequence from these templates as if they were translation examples. This match sequence is given in (2.40).

$$\begin{aligned} \text{at least } X^1_{Num-Card} \text{ book+Noun} &\leftrightarrow \text{en az } Y^1_{Num-Card} \text{ kitap+Noun} \\ \text{at least one+Num+Card } X^1_{Noun} &\leftrightarrow \text{en az bir } Y^1_{Noun} \end{aligned} \tag{2.39}$$

$$\begin{aligned} &\text{at least } (X^1_{Num-Card} \text{ book+Noun, one+Num+Card } X^1_{Noun}) \leftrightarrow \\ &\qquad \text{en az } (Y^1_{Num-Card} \text{ kitap+Noun, bir } Y^1_{Noun}) \end{aligned} \tag{2.40}$$

Regardless of the fact that the differences in the match sequence contain variables, we can learn the templates given below by running the similarity translation template learning algorithm.

$$\begin{aligned} \text{at least } X^1_{Num-Card\ Noun} &\leftrightarrow \text{en az } Y^1_{Num-Card\ Noun} \\ X^1_{Num-Card} \text{ book+Noun} &\leftrightarrow Y^1_{Num-Card} \text{ kitap+Noun} \\ \text{one+Num+Card } X^1_{Noun} &\leftrightarrow \text{bir } Y^1_{Noun} \end{aligned} \tag{2.41}$$

The user should note that learning translation templates from previously learned ones may yield three non-atomic templates. This was not possible when templates were extracted from translation examples.

While learning templates from previously learned templates, the constituents of a difference D_i may contain both variables and non variables. In that case, if we are going to learn a similarity translation template, we should expand the type labels of the variables in the constituents $D_{i,a}$ and $D_{i,b}$ in order to decide if epsilon insertion is necessary or not. An example of a difference D_i is given by

$$D_i\text{: } (X^1_{Verb} \text{ +PastSimp, } X^1_{Verb\ VERB-SUF-TENSE} \text{ +123SP}) \tag{2.42}$$

Even if both of the difference constituents contain two tokens, an epsilon insertion will turn out to be necessary if the variable type labels are expanded as shown in

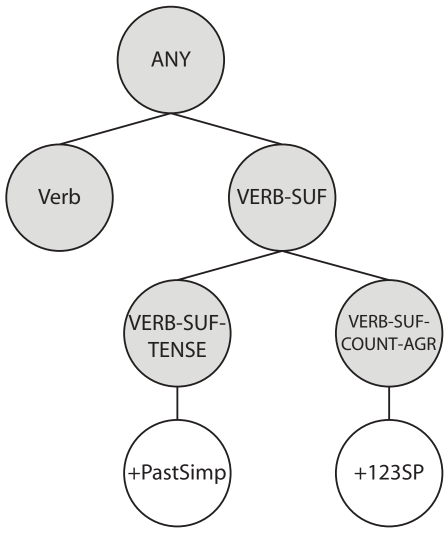


Figure 2.4: A Section of the English Type Lattice.

2.43.

$$\begin{array}{lll} \text{Verb} & \text{+PastSimp} & \\ \text{Verb} & \text{VERB-SUF-TENSE} & \text{+123SP} \end{array} \tag{2.43}$$

Now, there are obviously three posible epsilon insertion points for $D_{i,a}$ as shown below:

$$\begin{array}{lcl} (\epsilon \text{ Verb +PastSimp} & , & \text{Verb VERB-SUF-TENSE +123SP}) \\ (\text{Verb } \epsilon \text{ +PastSimp} & , & \text{Verb VERB-SUF-TENSE +123SP}) \\ (\text{Verb +PastSimp } \epsilon & , & \text{Verb VERB-SUF-TENSE +123SP}) \end{array} \tag{2.44}$$

Since an epsilon insertion will take place, we should be able to calculate the distances between type labels, in order to calculate the generalization scores of epsilon insertion possibilities. The Figure 2.4 provides the section of the English lattice that is necessary to solve the epsilon insertion problem.

$$\begin{aligned}
genScore_1 &= minDist(\epsilon,\ \text{Verb}) + minDist(\text{Verb, VERB-SUF-TENSE}) \\
&+ minDist(\text{+PastSimp, +123SP}) \\
&= 2+3+4=9 \\
genScore_2 &= minDist(\text{Verb, Verb}) + minDist(\epsilon,\ \text{VERB-SUF-TENSE}) \\
&+ minDist(\text{+PastSimp, +123SP}) \\
&= 0+2+4=6 \\
genScore_3 &= minDist(\text{Verb, Verb}) + minDist(\text{+PastSimp, VERB-SUF-TENSE}) \\
&+ minDist(\epsilon,\ \text{+123SP}) \\
&= 0+1+2=3
\end{aligned}$$

Since the third epsilon insertion point possibility has the lowest generalization score, it is chosen as the most appropriate one. As a result, the variable that replaces the difference in 2.42 is determined as $X_{Verb\ VERB-SUF-TENSE\ nullor(123SP)}$.

2.5 Confidence Factor Assignment

The translation templates generated during the learning phase, are stored in the file system, in order to be later used in the translation phase. Although the translations of some sentences submitted by the system user can be given using a single template[2], vast amounts of the translations are done using a combination of more than one translation template. During the translation phase, in order to translate a given sentence from the source language to the target one, a parse tree of templates are generated by the translation algorithm.

For most of the inputs, there will be multiple translation results. This is due to the fact that if the learned templates are general enough and numerous, there may exist multiple parse trees that can be used to translate the input phrase. Another factor that increases the number of results is the morphological ambiguities faced when converting the input from the surface-level to an equivalent lexical-level

[2]If the phrase submitted by the user and its translation exists in the translation examples file that is used to train the system, an atomic template that reflects this fact must have been learned.

representation.

This multiplicity of results is equivalent to that of a search engine. In order to increase the retrieval precision at the top ranks, a search engine fetching multiple results sorts them according to a criteria. The method here is to list the best results, in terms of relevance to the user query, at the top.

Similarly, in our system each translation result is assigned a confidence value and the results are then sorted in decreasing order of these values. The confidence value of a translation result is calculated as the multiplication of the confidence factors assigned to each template that is a node in the parse tree built in that particular translation [29].

Since the translation is bidirectional, each translation template is associated with not a single confidence factor, but with a pair of confidence factors. Then the first confidence factor is used for the translations from *language 1* to *language 2*, while the second one is used for the translations in the reverse direction.

A confidence factor is calculated as

$$confidence\ factor = \frac{N1}{N1 + N2} \tag{2.45}$$

where;

- N1 is the number of translation examples containing substrings on both sides that matches the template.
- N2 is the number of translation examples containing a substring only on the source language side that matches the template.

For example, assume that the translation examples file contains only the four examples below.

1. red+Adj hair+Noun +Sg ↔
   kızıl+Adj saç+Noun +A3sg +Pnon +Nom

2. red+Adj house+Noun +Sg ↔
   kırmızı+Adj ev+Noun +A3sg +Pnon +Nom

3. red+Adj ↔ kırmızı+Adj

4. long+Adj red+Adj hair+Noun +Sg ↔
   uzun+Adj kızıl+Adj saç+Noun +A3sg +Pnon +Nom

In order to assign the first confidence factor, which is to be used in English to Turkish translations, to a translation template such as

$$\text{red+Adj } X^1_{Noun} \leftrightarrow \text{kırmızı } Y^1_{Noun} \tag{2.46}$$

each translation example has to be evaluated individually. Initially both N1 and N2 are initialized to 0. The 1st example has a substring on its left side, "red+Adj hair+Noun", that matches the left side of the translation template. But there is no substring on the right that matches the template. So, N2 is incremented by 1.

Similarly, the 2nd example matches the translation template on the left hand side and it also has a substring on the right, "kırmızı+Adj ev+Noun", that matches the right hand side of the template. So N1 is 1.

The 3rd example does not match the template on either side, so N1 nd N2 remain unchanged.

The 4th example, like the first one, matches only on the left hand side; therefore, N2 is incremented to 2.

As a result, the English to Turkish confidence factor becomes $\frac{1}{1+2} = 0.33$. The reader can verify that the Turkish to English confidence factor becomes 1.0 using the same approach since $N1 = 1$ and $N2 = 0$ for that case.

While we are assigning a confidence factor to a template, we are actually approximating the ratio of the times a phrase matched by the source language side of the template is translated to a phrase matching the target language side of the template, to the total number of times such a phrase in the source language is ever translated.

If there are one or more type labels in a variable that are marked as *nullor*, then all possible type patterns for that variable should be considered while calculating the confidence factors.

2.6 Using Templates in Translation

The job of the learning component is over, when the templates are learned from the translation examples in the bilingual corpus file. When the user enters a phrase to the system in order to retrieve its translation, the translation component is responsible for handling this task.

Translation component first parses the input phrase using a slightly modified implementation [13] of the Earley parsing algorithm [19]. The Earley parser uses the learned translation templates as its grammatical rules. Since the templates are type associated, type checking is also performed by the translation component.

Parsing becomes successful if at least one parse tree can be built using a subset of the translation templates in the system. Usually, the parsing algorithm produces multiple parse trees, each representing a translation result. Then a translation result is produced merely by substituting each child template with the corresponding variable in the parent template, in a recursive fashion.

The generated results may be identical, as there may be multiple ways of reaching to the same translation result, or may be distinct. Some of those results will be incorrect semantically or syntactically due to the inappropriate generalizations during template learning. But, hopefully some correct translation results will also be generated.

For example, assume that the user wants to translate the phrase

"the plane was flying", (2.47)

which can be represented in the lexical form as

$$\text{the+Det +Def +SP plane+Noun +Sg be+Verb +PastSimp +Sg fly+Verb +Prog.} \tag{2.48}$$

Suppose that the known translation templates are as follows:

$$\begin{aligned}
&1 : \text{the+Det +Def +SP } X^1_{Noun\ Sg} \text{ be+Verb +PastSimp +Sg } X^2_{Verb} \text{ +Prog} \leftrightarrow \\
&\qquad Y^1_{Noun\ A3sg\ Pnon\ Nom}\ Y^2_{Verb} \text{ +Pos +Prog1 +Past +A3sg} \\
&2 : \text{plane+Noun +Sg} \leftrightarrow \text{uçak+Noun +A3sg +Pnon +Nom} \\
&3 : \text{plane+Noun +Sg} \leftrightarrow \text{düzlem+Noun +A3sg +Pnon +Nom} \\
&4 : \text{fly+Verb} \leftrightarrow \text{uç+Verb}
\end{aligned} \tag{2.49}$$

where the associated English to Turkish confidence factors are 0.9, 0.8, 0.2 and 1.0, respectively.

When the parsing algorithm runs on the lexical-level form of the input phrase, the parse trees in Figure 2.5 will be returned.

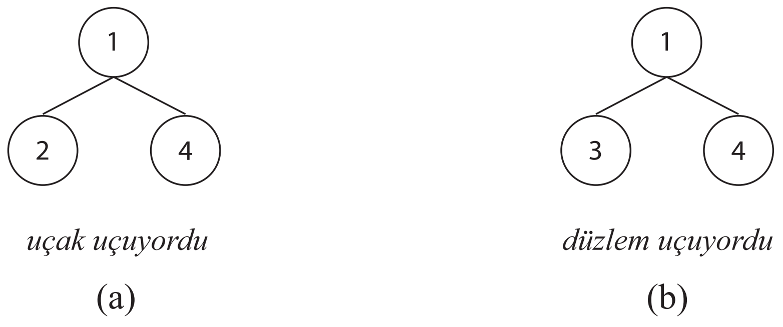


Figure 2.5: Translation Results for the Phrase (2.47). Note that the corresponding translation results are given in the surface-level form.

When the translation is over, the results will be presented to the user. Before doing this, the results will be sorted in decreasing order of confidence values. As stated earlier, the confidence value of a translation result is determined as the product of the translation templates used in generating that translation result. The confidence value of the translation result in Figure 2.5(a) is $0.9 \times 0.8 \times 1.0 = 0.72$, while the confidence value of the translation result in Figure 2.5(b) is $0.9 \times 0.2 \times 1.0 = 0.18$. Therefore, the first translation result will be ranked above the second one. This complies with our expectations, as semantically the first translation result is correct, while the second one is not.

Chapter 3

System Architecture

This chapter describes the interactions among various components that make up our translation system. While some minor components will be covered in this chapter, remaining major components, such as User Evaluation Interface and the Turkish Morphological Disambiguator will only be mentioned here briefly, as the following chapters cover them in detail.

Due to the portability advantage and string manipulation capabilities offered, we developed our system using the Java Programming Language[1] (Java SE 6 SDK). Instead of developing the whole application as a single .jar package, we divided it into several chunks, all of which can be used as independent applications, or can be included in other projects as libraries.

Figure 3.1 depicts the interactions among the components of our system. Workings of the system can be divided into two phases. In the first phase, the system is trained using a bilingual aligned corpus. This traning corpus contains bilingual translation examples in their lexical form. Training of the system finishes when the *Learning Component*, principles of which are described in detail in Chapter 2, writes the translation templates it has learned, into a file. In the first phase, the user is passive, i.e., the learning process is completed without any user interaction.

[1]Java homepage is at http://java.sun.com.

The second phase uses the translation templates extracted in the previous phase, in order to translate the natural language phrases input from the user. Unlike the previous one, the translation phase is interactive, i.e., the *Translation Component* asks the user to enter a phrase in either of the Turkish or English languages, and after performing the translation, returns the results back to the user, and waits for the next input (see Section 2.6).

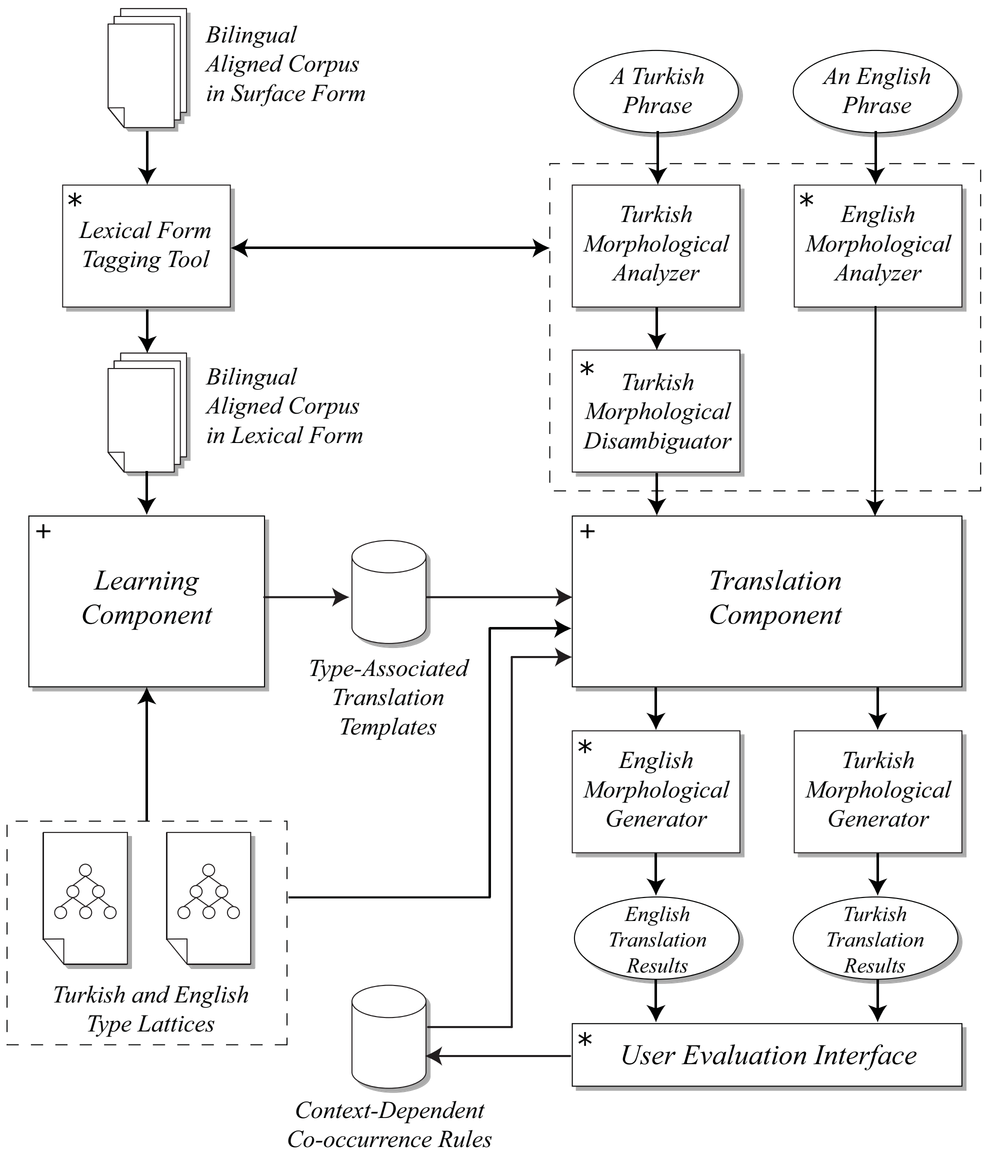
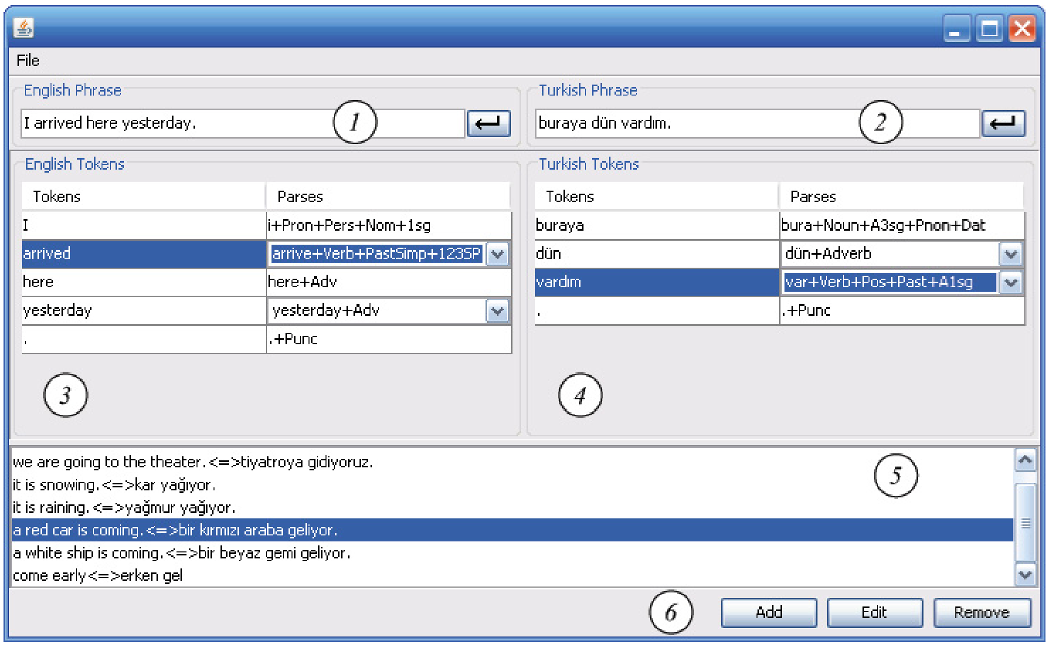

Bilingual Aligned Corpus in Surface Form
A Turkish Phrase
An English Phrase
* Lexical Form Tagging Tool
Turkish Morphological Analyzer
* English Morphological Analyzer
Bilingual Aligned Corpus in Lexical Form
* Turkish Morphological Disambiguator
+ Learning Component
+ Translation Component
Type-Associated Translation Templates
* English Morphological Generator
Turkish Morphological Generator
English Translation Results
Turkish Translation Results
Turkish and English Type Lattices
* User Evaluation Interface
Context-Dependent Co-occurrence Rules

Figure 3.1: A detailed view of the system components. The components developed during this thesis study are marked with a star sign (*), and the components modified during this thesis study are marked with a plus sign (+) in the upper left corner.

3.1 Lexical-Form Tagging Tool

In order to extract some translation templates, the Learning Component takes a bilingual corpus as input. This corpus has to be in the lexical-form, as using the lexical-forms of the translation examples enables the system to learn more useful templates when compared to using marely the surface-forms. Manually converting translation examples in a corpus, from their more natural surface-forms into lexical-forms, without using any software tool, would be an inefficient and error-prone task. Therefore, we have developed a tagging tool that simplifies the conversion process.

The user interface of the tagging tool is given in Figure 3.2. The "File" menu provides options for reading a corpus in surface-form from a file and converting it into lexical-form, saving a processed corpus, creating a new blank corpus and opening a previously saved corpus.



I arrived here yesterday.

buraya dün vardım.

Tokens	Parses
I	i+Pron+Pers+Nom+1sg
arrived	arrive+Verb+PastSimp+123SP
here	here+Adv
yesterday	yesterday+Adv
.	.+Punc

Tokens	Parses
buraya	bura+Noun+A3sg+Pnon+Dat
dün	dün+Adverb
vardım	var+Verb+Pos+Past+A1sg
.	.+Punc

```
we are going to the theater.<=>tiyatroya gidiyoruz.
it is snowing.<=>kar yağıyor.
it is raining.<=>yağmur yağıyor.
a red car is coming.<=>bir kırmızı araba geliyor.
a white ship is coming.<=>bir beyaz gemi geliyor.
come early<=>erken gel
```

Figure 3.2: Lexical-Form Tagging Tool. **(1)-(2)** The English and Turkish text fields used to enter a new translation example. **(3)** The table that shows each English token and the associated lexical-level representation. **(4)** The table that shows each Turkish token and the associated lexical-level representation. **(5)** The area that lists the translation examples existing in the corpus. **(6)** Control buttons that are used to add, edit and remove translation examples.

The user can also enter new translation examples interactively by filling the text fields at the top of the window. When a new translation example is entered, the tagging tool determines the morphological parses associated with each token in the left and right constituents of that example. Then the tokens and the corresponding parses are presented in a table format.

Morphological parsing of the tokens are done by calling the methods provided by the English and Turkish morphological analyzer libraries. If more than one morphological parses exist for a certain token, i.e., the token is morphologically ambiguous, then those parses are shown in a drop-down list, from which the user can then select the correct parse. Resolving the morphological ambiguities by selecting the correct parse for each token should be done by the user, as the translation examples used while tranining the system have to be unambiguous.

3.2 Morphological Analyzers

Morphological analysis[2] is the process of breaking a given word into its morphemes. Morphological analysis is closely related to *stemming*, as this problem can be reduced to the former one, i.e., using a morphological analyzer, we can directly obtain the root of an input word, by inspecting the morphological parse produced for it by the analyzer[3].

Morphological generator does the opposite of morphological analyzer by converting a given morphological parse into its surface form. In our system, the morphological analyzers are bundled with their corresponding generators. Following subsections discuss the morphological anaylzers for Turkish and English languages used in our system.

[2]Used interchangeably with *morphological parsing*.

[3]If the word is morphologically ambiguous, then this may prevent us from identifying the stem correctly. In that case, some means of disambiguation will be necessary.

3.2.1 Turkish Morphological Analyzer

In our translation system, we are using the version 2.1.13 (October 25, 2002) of PC-KIMMO morphological analyzer engine [2], which is implemented in the C programming language. PC-KIMMO is a general purpose processor for two-level morphological descriptions, and its source code is freely available for non-commercial purposes. The program is designed to recognize (parse) and generate (produce) words using a two level language description. The recognition function converts a given word from surface-level form, into lexical-level form, while the generation function works in the reverse direction.

The two-level description of the Turkish language used by PC-KIMMO was previously developed by Oflazer [25]. Also, several modifications have been introduced to this description recently [17], such as re-organizing the output format into a more standard one and changing the internal encoding to ISO-8859-9, which covers Turkish specific letters. The Turkish lattice structure used by the translation system is provided in [13]. Some recognition examples for the morphological analyzer are given in Table 3.1.

Table 3.1: Some Recognition Samples for the Turkish Morphological Analyzer.

Token	**Parse(s)**
ev	ev+Noun+A3sg+Pnon+Nom
evden	ev+Noun+A3sg+Pnon+Abl
evimden	ev+Noun+A3sg+P1sg+Abl
evi	ev+Noun+A3sg+P3sg+Nom ev+Noun+A3sg+Pnon+Acc
geldim	gel+Verb+Pos+Past+A1sg
veriyordum	ver+Verb+Pos+Prog1+Past+A1sg
hızlı	hız+Noun+A3sg+Pnon+Nom^DB+Adj+With

In order to be able to use the function calls provided by PC-KIMMO, we have developed a Java language interface to it. This interface provides recognition and generation methods customized for easy use in the Java environment, and more general methods that can be used to run any command that the PC-KIMMO command line application accepts. It also supplies a java command line emulator application, by which the user can interact with the original PC-KIMMO command line interface. The Java interface to PC-KIMMO is designed to be independently used in other projects as well, hence provided as a seperate .jar package.

When the user enters an input phrase in Turkish for translation, this phrase is first split into its tokens, such as words, numerals and punctuation marks[4]. Then on each token the recognition function of the Turkish morphological analyzer is run. Due to the morphological ambiguities of Turkish, the recognizer generally returns multiple morphological parse results, which are then fed into the morphological disambiguator. Likewise, when an English phrase is translated into Turkish, the output of the Translation Component is converted into surface-form by the generator function before being presented to the user.

3.2.2 English Morphological Analyzer

Previous version of our translation system used the online Xerox morphological analyzer[5] to convert English words from surface to lexical-level forms. Basically, when need for the morphological parse of a new word arose, it queried the online morphological analyzer. The result was stored in a cache file, which was used as a lookup table to increase the retrieval speed of the morphological parses. This cache file was also used in the morphological generation step which converts the lexical-level representations of the words back to their surface-level representations.

Accessing the online Xerox morphological analyzer for each distinct English word could only serve as a short-term solution, as the online querying approach requires Internet connectivity and is time consuming. Also searching the morphological parse of a word in a non-indexed file was inefficient.

[4]The tokenization service is not built in PC-KIMMO; it is provided by the *Supervised Tagger* library, which will be discussed in Chapter 4.

[5]Accessible at http://www.xrce.xerox.com/competencies/content-analysis/demos/english.

The reason for using the Xerox morphological analyzer was, that it produces morphological parses similar in many aspects to the parses produced by the Turkish morphological analyzer. Other options, such as the two-level specification of English available for PC-KIMMO, provide a very different formatting approach for morphological parses. Therefore, using such a morphological analyzer would add additional complexities in the template learning mechanisms.

In order to overcome the issues mentioned above, we have developed an English morphological analyzer. Although there are some changes, the parse formatting of our analyzer is very similar to that of the online Xerox morphological analyzer and the Turkish morphological analyzer. Sample outputs of the English morphological analyzer are given in Table 3.2.

Table 3.2: Some Recognition Samples for the English Morphological Analyzer.

Token	**Parse(s)**
come	come+Verb+Prog^DB+Adj+Zero come+Verb+Prog^DB+Noun+Zero+Sg come+Verb+Prog
there	there+Adv there+Pron there+Pron+Nom+3SP
umbrella	umbrella+Noun+Sg
she	she+Pron+Pers+Nom+3sg
their	they+Pron+Pers+Gen+3pl

did	do+Verb+PastSimp+123SP
	do+Aux+PastSimp+123SP

belongings	belong+Verb+Prog^DB+Noun+Zero+Pl
bought	buy+Verb+PastPerf+123SP^DB+Adj+Zero buy+Verb+PastSimp+123SP buy+Verb+PastPerf+123SP
teacher	teach+Verb+Inf^DB+Noun+Er+Sg
goes	go+Verb+Pres+3sg
substitution	substitute+Verb+Inf^DB+Noun+Ion+Sg
freely	free+Adj^DB+Adv+Ly

English is a weakly inflected language. Therefore in our morphological analyzer, the only inflectional suffixes used are verb tense suffixes (-ed, -s, -ing), noun suffixes (-s, -'s), and adjective suffixes (-er, -est). On the other hand, English uses derivational suffixes extensively just like Turkish. Hence, our morphological analyzer recognizes 51 commonly used derivational suffixes (see Appendix B). Also there is a special suffix named ZERO, which is used when derivation is done without affecting the surface-form. For example, most of the verbs in past perfect tense, can also be used as adjectives. In this case, we denote the derivation from verb to adjective with the ZERO suffix.

Root words and exceptional inflection and derivations are kept in lexicon files. Words are categorized into lexicon files according to their part-of-speech. The number of entries in each lexicon file is given in Table 3.3.

Table 3.3: Number of Root Words and Exceptional Cases in Each Lexicon.

Lexicon File	**Number of Entries**
Noun	10796
Verb	3859
Adjective	3311
Adverb	375
Abbreviation	114
Other	538
Total	**18993**

In addition to suffixes, English also uses prefixes which are appended to the beginnings of words. The current version of our morphological analyzer does not recognize prefixes. Therefore, prefix-derived forms of the words have to be added to the lexicon files, as if they were root words.

As we now use a different morphological analyzer for English, the lattice structure used in the previous version of the system has to be modified. The complete list of the nodes in the new English Lattice, and the relationships between those nodes are given in Appendix C.

3.3 Turkish Morphological Disambiguator

As mentioned previously, Turkish morphology is highly ambiguous and during the morphological analysis, multiple parses will often be associated with a given input token. When multiple ambiguous tokens exist, the Translation Component has to be run on each possible combination of the lexical representations of the tokens in the input phrase. This does not reduce the *recall* of the translation system, as the correct combination will always be tested among many incorrect combinations. But the problem is that the translation *precision* will be affected to a great extent, as the incorrect morphological parse combinations will increase the number of wrong translations.

Morphological disambiguation is the process of selecting the most suitable morphological parse for a given word, from the set of parses that is assigned to that word by the morphological analyzer. Unlike the ideal case, sometimes the disambiguator cannot select a single parse; in this case it should eliminate as much wrong parse as it can.

In situations, where syntactic and morphological information derived from a word in a given text falls short of correctly identifing the lexical class of it, further evidence from the surrounding words can usually be utilized to disambiguate it.

As shown in Figure 3.1, the Turkish Morphological Disambiguator sits between the Turkish morphological analyzer and the Translation Component in our system. The Disambiguator Component is provided by the *Supervised Tagger* library which will be disscussed later in Chapter 4.

3.4 User Evaluation Interface

One of the new enhancements we propose for our translation system is a user feedback mechanism. When the translation system generates multiple results, either due to the morphological ambiguities discussed above or multiple translation template combination options for translation, the results are presented to the user in descending order of confidence values.

In the previous versions of the system, during the translation phase, the user had no effect on the confidence values assigned to each result, hence on the presentation order of the results. In order to reflect his preference into the results ordering, a user had to enter more translation examples and rerun the learning component, which consumes computation resources and takes time. Moreover, in a realistic situation, it will be impossible for a user to estimate the number of examples to add, that will adjust the ordering of the results to the desired configuration.

We propose a new method of incorporating user feedback into the result ordering mechanism. By evaluating the translation results generated, the user can teach his preferences to the system. From the evaluation data, the system extracts template co-occurrence rules, which specify aggregate confidence factors for certain template configurations. The extracted rules are then kept in the file system to be used in later translations. Chapter 5 discusses the user feedback mechanisms in detail.

Chapter 4

Morphological Disambiguation

Morphological disambiguation is the process of eliminating inappropriate parses assigned to a word by the morphological analyzer. In other words, a morphological disambiguator uses the output of the morphological analyzer, and for each word in the text, tries to select a single morphological parse, from the set of parses assigned to that word. When eliminating all but one of the parses is not possible, we expect, that the disambiguator selects a subset of the parses of minimum size.

Structural information obtained from a word in a given text can sometimes be sufficient to correctly disambiguate it. For example, if the first letter of a word is written in upper case, but the word is not the first one in the sentence, then that word is most probably a proper noun.

Usually, however, we cannot simply identify the lexical category of a word by inspecting just its own syntax and morphological structure. Fortunately, much useful evidence can be collected from the context that the word lies in, i.e., most of the time, the morphological information gathered from neighboring words can successfully be utilized to disambiguate the target word.

A problem, that is closely related to morphological disambiguation, is part-of-speech (POS) tagging. In part-of-speech tagging, the aim is to find the correct lexical category of a given word. POS tagging can be reduced to the problem of morphological disambiguation, as finding out the correct parse of a word, will lead us to determine its lexical category as well.

Although 100% accurate POS tagging or morphological disambiguation is practically impossible, highly accurate systems for English are available. The weak inflectional morphology of English, helps increase the effectiveness of those systems. For Turkish on the other hand, POS tagging and morphological disambiguation are much more complicated processes in general. This is due to the inherent morphological level ambiguity of the language. Agglutinative nature of Turkish makes the number of morphological parses for each word larger than that of English. According to [20], about 80% of all words are morphologically ambiguous in Turkish. An obvious example is the word “kitabın”:

Kitabın eski.	↔	Your book is old.
Kitabın kapağı maviydi.	↔	The cover of the book was blue.

Here the ambiguity is due to the phonetic similarity of the genitive suffix *-in* and the second singular possessive suffix *-(n)in*. Similarly, nominals with the accusative suffix *-(y)i* and the third singular possessive suffix *-(s)i*, may have the same surface form if the root ends with a consonant. Another kind of ambiguity arises when the root of one word is a prefix string of the root of another word, and the shorter root is appended a suffix which causes the two words to surface to the same string. A typical example is the word “altın”. For this word, the morphological analyzer gives the following parses:

1. altın+Adj
2. altın+Noun+A3sg+Pnon+Nom
3. altı+Num+Card^DB+Noun+Zero+A3sg+P2sg+Nom
4. alt+Noun+A3sg+Pnon+Gen
5. alt+Noun+A3sg+P2sg+Nom

The problem in this case is, that both “alt” (sub, below, lower) and “altı” (six) are prefixes of the word “altın” (gold). In fact, this is one of the most common kinds of morphological ambiguities observed in Turkish [28].

Automatic disambiguation is very important for high level NLP applications, such as our machine translation system, as the performance of this kind of systems tend to degrade, when too many words in an input are ambiguous. As an example, assume that the user wants to translate the noun phrase

“*yeni gelişme*”

into English. When the Turkish morphological analyzer is run on the words in this phrase, the results in the following table are returned:

Table 4.1: Morphological Analysis Results for the Noun Phrase: “*yeni gelişme*”.

Token	**Parses**
yeni	yen+Noun+A3sg+P3sg+Nom
	yen+Noun+A3sg+Pnon+Acc
	yeni+Adj
gelişme	geliş+Verb+Neg+Imp+A2sg
	geliş+Verb+Pos^DB+Noun+Inf2+A3sg+Pnon+Nom

In order to translate the input phrase into English, we should take the Cartesian product of the two parse sets, and feed all elements of this set into the translation component. For our example, there are a total of $3 \times 2 = 6$ elements in the Cartesian product set. Therefore, the translation algorithm will run a total of 6 times, consuming precious time. Also, only one of the six elements fed into translation algorithm will be correct. If the translation templates used by the system are general enough, we should expect that the incorrect elements will cause invalid translation results, and will probably degrade the translation precision.

In order to save time and increase the translation precision, we have developed a morphological disambiguator for Turkish. This chapter provides an elaborate description of this tool.

4.1 Related Works

There are two broad categories of POS tagging algorithms which are rule-based taggers and stochastic taggers [19]. Rule based taggers contain a database of hand-crafted rules that are designed to minimize ambiguity when applied in a certain order on each word in the text. Statistical POS taggers (also known as stochastic taggers), use a training corpus to calculate the likelihood of co-occurrence of all ordered pairs of tags. By training a probabilistic model such as Hidden Markov Model (HMM), the tagger tries to disambiguate any given new text.

The earliest algorithms for automatic POS tagging were the rule-based ones. The tagger that was an aid in tagging the famous Brown Corpus was a rule-based one. This tagger, named TAGGIT, was able to disambiguate 77% of the Brown corpus, the remaining parts of which were tagged manually [9].

Stochastic techniques have proven to be more successful compared to pure rule-based ones. Church at. al. [8] presented a stochastic method that achieved over 95% accuracy. Also Cutting at. al. [10] presented a POS tagger based on a HMM that enables robust and accurate tagging with only a lexicon and some unlabeled training text requirements. According to the authors accuracy exceeds 96%.

Brill [3] introduced a rule based POS tagger which used a transformation based method that learns its rules from a training corpus. Brill tagger has performance comparable to the statistical taggers stated above. Unfortunately, Brill tagging is not directly applicable to agglutinative languages such as Turkish [27].

In [1], Altıntaş et al. introduce a stemming method for Turkish. After the morphological analysis step, the best stem is determined using stem-length information collected from a disambiguated corpus. The stem whose length is closest to the average stem-length of the corpus is selected. If there is more than one result with the same length, the part-of-speech information of the stem is considered, and the stem that belongs to the more common lexical category is selected.

Current trend in morphological disambiguation and POS tagging is blending machine learning techniques and statistic methods into rule based approaches. Oflazer and Kuruöz [26, 20], developed a POS tagger that uses local neighborhood constraints, heuristics and limited amount of statistical information. Oflazer and Tür [27] developed a system that combines corpus independent, linguistically motivated hand-crafted constraint rules, constraint rules that are learned via unsupervised learning from a training corpus, and additional statistical information from the corpus to be morphologically disambiguated.

4.2 A Morphological Disambiguator for Turkish

As a part of this thesis, we developed a rule-based morphological disambiguation tool which is based on the previous work in [2, 7]. Our tool differs from the original one with its easy to use user interface, and more elastic rule specification format, which is fully compatible with the output format of the new two-level description of Turkish[17] prepared for PC-KIMMO[2]. Our morphological disambiguator is a part of the *Supervised Tagger* package.

Supervised Tagger package can be used on its own, as it contains an application with an easy to use graphical user interface (GUI), that disambiguates input texts. It can also be used as a library in higher level NLP applications, as we have done in our example-based translation system. The "supervised" in the name of the package is due to the fact, that the user can supervise the tagging process in the half-automatic mode provided by the GUI application. When used as a library, Supervised Tagger works in full-automatic mode.

Supervised Tagger not only provides morphological disambiguation function, but also supplies a tokenizer, collocation recognizer and an unknown word recognizer. It operates as follows on an input text: The text is first divided into its tokens. Then the morphological analyzer is sequentially run on each token and a list of parses is associated with each word. Then the unknown word recognizer is run on the tokens, for which the morphological analyzer has returned

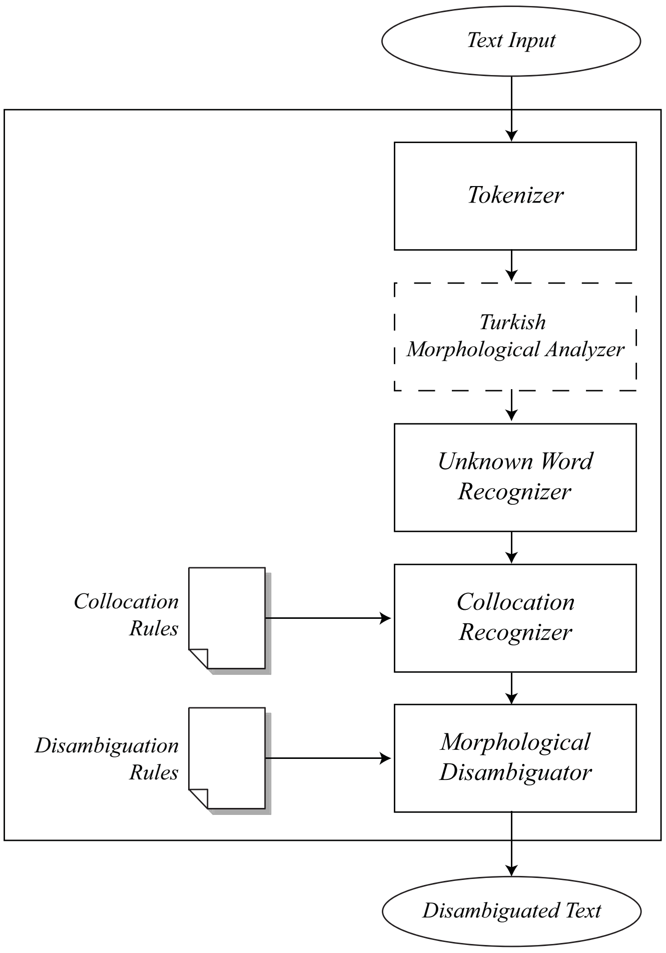


Figure 4.1: The Operation of Supervised Tagger. Note that the Turkish morphological analyzer is not included in the Supervised Tagger package.

an empty list. After that, the collocations, word sequences that constitute some special meaning when used together, are detected by the collocation recognizer and packed into composite tokens. Lastly the morphological disambiguator is run on the token sequence, which detects and eliminates improper morphological parses using context sensitive rules.

The operation of Supervised Tagger is depicted in Figure 4.1. In the following subsections, we describe each module of Supervised Tagger that have been mentioned above.

4.2.1 Tokenizer

Collocation recognition and disambiguation rules are applied to an input text that is represented as a sequence of tokens. Therefore, tokenization must be applied to an input text first. Supervised Tagger detects token boundaries mainly using white-space characters and punctuation marks. The tokenizer is also smart enough in detecting some composite forms that use punctuation marks, such as the real and ordinal numbers. Token types given in Table 4.2 are properly recognized by the tokenizer.

Table 4.2: Token Types Recognized by the Tokenizer.

Type	**Example**
Ordinary words	evde, geldim, bugün, etc.
Numeric structures	334, 2,5, %10'unda, 1., 10:45'te, etc.
Punctuations	'.', '...', ',', '!', '?', ':', ';', etc.

Detecting sentence boundaries to help the disambiguation process is the job of the tokenizer. Tokenizer detects the beginning and ending of sentences by marking sentence delimiting punctuation marks. Unfortunately successfully detecting sentence boundaries is not a trivial task in its own. Some of the punctuation marks that are used to delimit sentences ('.', '!', '?', ':', '...'), are also commonly used as special markers inside tokens. E.g., period is used in abbreviations and ordinal numbers; and colon is used in time formats. Handling of ordinal and real numbers and some other forms containing numbers and punctuation marks are both handled in this step. Sentence boundaries detected during tokenization are later refined in the collocation recognition step, as the abbreviations are handled not in tokenization but in the collocation recognition step. Tokens of some sample sentences that contain numeric structures are given in Table 4.3.

4.2.2 Unknown Word Recognizer

After the morphological analysis there may remain some tokens that are not assigned any parses such as some foreign proper nouns or mistyped words. These

Table 4.3: Tokenization Examples for Numerical Structures.

Sentence	Tokens
buraya 3:40'ta geldim.	"buraya", "3:40'ta", "geldim", "."
yarismada 1. olmuşum.	"yarişmada", "1.", "olmuşum", "."
1,5 metre yüksekliğindeydi.	"1,5", "metre", "yüksekliğindeydi", "."

tokens are currently handled by the unknown token recognizer module. This unknown token recognizer also uses PC-KIMMO as a backend, but applies some root substitution methods that use phonetical rules of Turkish, in order to find suitable morphological parses for unknown tokens.

As a simple example we can give the token "bienalde" (at the biennale). The word "bienal" is not included in the lexicon of the morphological analyzer, so it is an unknown token. In order to find the correct parses of this token, the RECOGNIZE-UNKNOWN-TOKEN procedure is executed (see Algorithm 3).

```
RECOGNIZE-UNKNOWN-TOKEN($token$)
 1:  $n = length(token)$
 2:  $parse\_list \leftarrow \emptyset$
 3:  for $i = x$ to $n$, where $x$ is the position of the first wovel in the token do
 4:      $stem \leftarrow$ substring of $token$ from character 1 to $i$, both inclusive
 5:      $suffix \leftarrow$ substring of $token$ from character $i+1$ to $n$, both inclusive
 6:      choose a proper pseudostem
 7:      $result \leftarrow$ MORPHOLOGICAL-ANALSIS($pseudostem + suffix$)
 8:      replace $pseudostem$ with $stem$ in each $parse \in result$
 9:      add all $parse \in result$ to $parse\_list$
10:  end for
11:
12:  return $parse\_list$
```

Algorithm 3: RECOGNIZE-UNKNOWN-TOKEN. Recognizes the tokens unknown by the morphological analyzer.

The pseudostem selection is the most critical part in this algorithm. The pseudostem is selected using some phonetic attributes of the stem and suffix, and it is basically done as a lookup from a table containing preselected tokens that are guaranteed to be known by the morphologic analyzer. The results of the unknown token recognizer for the token "bienalde" are given below:

1. bienal+Noun+A3sg+Pnon+Loc
2. bienald+Noun+A3sg+Pnon+Dat
3. bienald+Verb+Pos+Opt+A3sg
4. bienalde+Noun+A3sg+Pnon+Nom

Unknown abbreviations, foreign proper nouns, unknown verbs, inflected numeral forms unhandled by the morphological analyzer, etc. can be correctly handled by the unknown token recognizer.

4.2.3 Collocation Recognizer

The third module of Supervised Tagger, the collocation recognizer, takes the morphologically analyzed text as input, and tries to detect and combine certain lexicalized and non-lexicalized collocations in it. The need for such a processing arises from the fact that a group of words, when appeared subsequently in a sentence, may behave as a multiword construct with a totally or partially different function compared to its individual members in the sentence. A typical example is the construct "*gelir gelmez*":

$$\begin{array}{rcl} \text{O } \underline{\text{gelir}}. & \leftrightarrow & \text{He } \underline{\text{comes}}. \\ \text{O } \underline{\text{gelmez}}. & \leftrightarrow & \text{He } \underline{\text{does not come}}. \\ \text{O } \underline{\text{gelir gelmez}} \text{ ayrıldık.} & \leftrightarrow & \text{We left } \underline{\text{as soon as}} \text{ he } \underline{\text{comes}}. \end{array}$$

Here the words "gelir" and "gelmez", when used together, function in that sentence as an adverb, whereas the words are inflected verbs when considered individually. There are a number of other non-lexicalized forms which are in general in the form $w + xw + y$, where w is the duplicated string of a root and certain suffixes and x and y are possibly different sequences of other suffixes [27]. Examples of other non-lexicalized collocations supported are "*hızlı hızlı*", "*güzel mi güzel*", "*koşa koşa*", "*kapı kapı*", "*yaptı yapalı*" and "*şırıl şırıl*".

The collocation recognition is performed according to the rules given in the collocation rules file, which contains around 340 rules currently. XML (Extensible Markup Language) file format is chosen for collocation rules because of its flexibilities. An XSD (XML Schema Definition) file is also used during rule loading to check the validity of rules in the rules file. A collocation rule is a sequence of token constraints and an action statement. If the sequence of token constraints matches a sequence of tokens in the text that is analyzed, the action in the action statement is applied. An action statement provides a template, using which the collocation recognizer can combine the tokens in the matched sequence into a single composite token. For example, the rule that handles the collocation "*gelir gelmez*" is given below:

```
<collocationRule>
<constraint><parse>_R+Verb+Pos+Aor+A3sg</parse></constraint>
<constraint><parse>_R+Verb+Neg+Aor+A3sg</parse></constraint>
<action>%1 %2+Adverb+When</action>
</collocationRule>
```

In the rule above, the first constraint matches a token such as "*duyar*", "*yapar*", "*görür*", etc. and the second constraint matches tokens such as "*duymaz*", "*yapmaz*", "*görmez*". In both of the constraints, the stem of the token is represented with the variable "_R". In the constraints, the character "_" and the letter immediately following it, represents a variable. When the constraint matches a token, the stem of the token is stored in a data structure which uses this letter as a label. Since both constraints cause the stems of matching tokens to be stored in the same data structure labeled as "R", the stems must be identical. If they are not, then matching tokens will not be considered as a collocation.

The action statement is used to define a template according to which the collocation recognizer can combine the matched tokens into a composite token. Again, some special markers are used in the action statement. These markers begin with the character "%". If a number follows "%", then this number denotes a token sequence number. If a letter follows "%", then this denotes a stem stored in the previously mentioned data structure with the given letter used as a label.

The action statement of our example collocation rule

```
<action>%1 %2+Adverb+When</action>
```

declares the parse that should be assigned to the composite token when a sequence of tokens matches the constraint sequence of the rule. This parse, as declared by the action statement, contains readings of the first and the second tokens in the match sequence, seperated with a space character and followed by the tag "+Adverb+When".

A constraint does not always have to declare a parse to be matched, but also token readings can be matched. This kind of rules are especially used for detecting lexicalized collocations, as it is the case for the example rule given below:

```
<collocationRule>
<constraint><token>hiç</token></constraint>
<constraint><token>kimse</token></constraint>
<action>%1 %2+Pron</action>
</collocationRule>
```

This rule combines tokens that have readings "*hiç*" and "*kimse*" into a single token "*hiç kimse*" which is then labeled as pronoun. It is also possible to use regular expressions when writing token constraints. An examplary rule is given below:

```
<collocationRule>
<constraint><token matchType="regex">m[ös]</token></constraint>
<constraint><token>.</token></constraint>
<action>%1.+Adj</action>
</collocationRule>
```

This rule is an example to the collocation rules that detect abbreviations.

Two abbreviations, "*mö.*" and "*ms.*" are detected by this rule. Token matching by regular expressions is case sensitive while the ordinary token matching is case insensitive.

4.2.4 Morphological Disambiguator

Morphological analysis of a Turkish word usually returns multiple morphological parses. This ambiguity is due to the agglutinative nature of the language. Morphological disambiguator module of Supervised Tagger, using a set of context sensitive, hand-crafted rules, aims to reduce the number of parses associated with each word.

Disambiguation is performed using two types of disambiguation rules, namely **choose** and **delete** rules. These rules are applied only if a word is in the context specified by the rule. By being in the context, we mean that the surrounding words match the constraints of the rule. A disambiguation rule must target a token, i.e. the token that this rule aims to disambiguate primarily. A rule can also specify neighboring tokens, each described by an offset value, i.e., the relative position of the neighbor according to the target.

A high percentage of disambiguation rules in our system are ported from [32]. The difference is that our tagger uses a more expressive formatting for disambiguation rules when compared to the former work. Again, the XML file format is preferred for the disambiguation rules and an XSD file is supplied to check the validity of a newly added rule during rule loading. An examplary rule is given below:

```
<chooseRule>
<neighbour offset="-1"><parse>Dat</parse></neighbour>
<target><parse>düş+Verb</parse></target>
</chooseRule>
```

This rule is one of the many choose rules that are stored in the disambiguation rules file. Most of the choose rules in this file are motivated by the grammatical constraints of Turkish; so they are independent from the text category. When choose rules are applied to a certain word, if the constraints of the rule are satisfied, then the target token and its ambiguous neighbors are disambiguated at once. As an example, for the noun phrase "*çocuğun kitabı*", the morphological analyzer, by analyzing the words in the phrase independently, returns us the parses given in Table 4.4.

Table 4.4: Morphological Analysis Results for the Phrase: "*çocuğun kitabı*".

Token	**Parses**	
çocuğun	çocuk+Noun+A3sg+Pnon+Gen	
	çocuk+Noun+A3sg+P2sg+Nom	(correct)
kitabı	kitap+Noun+A3sg+Pnon+Acc	(correct)
	kitap+Noun+A3sg+P3sg+Nom	

A rule in our disambiguation rules file,

```
<chooseRule>
<neighbour offset="-1"><parse>A3sg+Gen</parse></neighbour>
<target><parse stemAllowed="false">Noun+P3sg</parse></target>
</chooseRule>
```

matches the noun phrase "*çocuğun kitabı*". When we apply this rule on the noun phrase, not only the word "*kitabı*" is disambiguated, but also the appropriate parse for its neighbor, "*çocuğun*", is chosen.

Another set of rules, called delete rules, are also used in the disambiguation process. Delete rules are mainly used to remove very rare parses of some common words. This set of rules only affect the word that is being disambiguated, and they work only in a unambiguous context. An examplary delete rule

```
<deleteRule>
<target><token>biz</token><parse>Noun</parse></target>
</deleteRule>
```

drops the infrequent noun parse of the word "biz" in favor of the very commonly used pronoun parse.

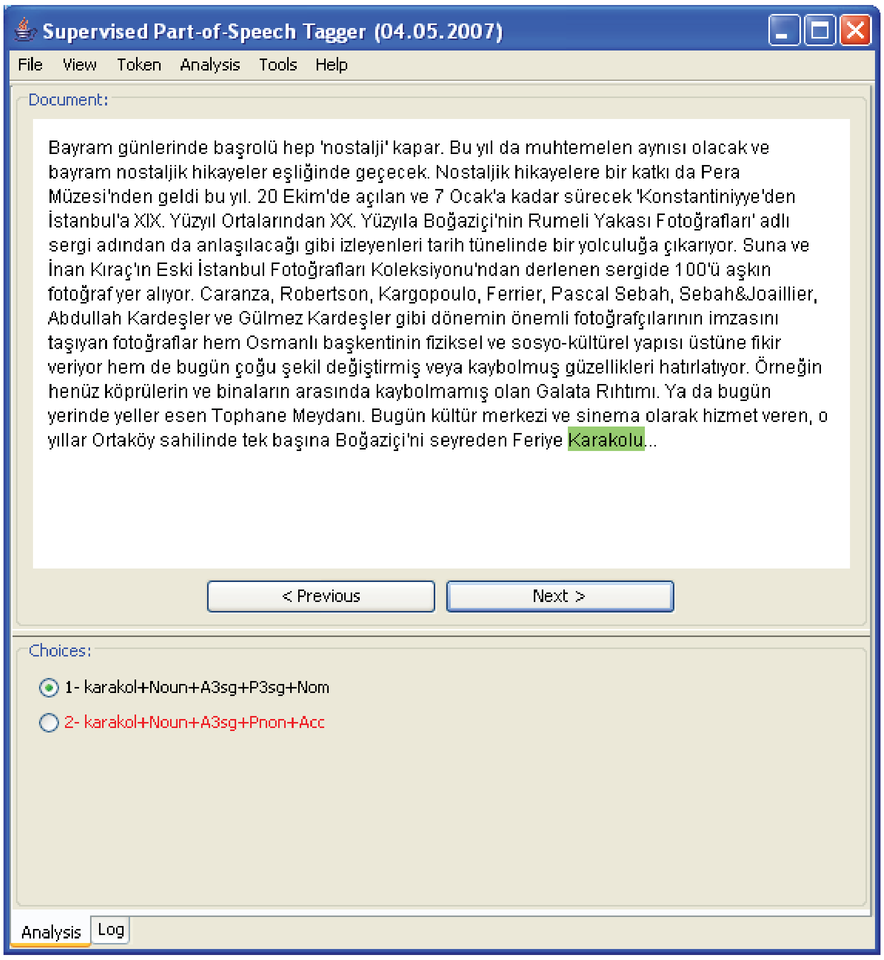
The rules in the disambiguation rules file are grouped according to their categories. They are also ordered according to their generalities; the more a rule is stricter (specific), the higher in the file it would appear. The order of the rules is very important, because if the ordering is wrong, then the disambiguation will produce more wrong results.

4.3 Morphological Annotation Tool

The developed morphological disambiguator is integrated with a graphical user-interface, so that it can be used as a morphological annotation tool. In fact, our test data that is used to evaluate the disambiguation performance is prepared using this annotation tool. A human expert can use this tool to morphologically annotate a corpus.

The main graphical user-interface window of the annotation tool is given in Figure 4.2. The user can load a file using the "File" menu and execute the disambiguation process using the "Analysis" menu. The user can annotate the text by the help of the disambiguator, and save the completely or partially tagged text. If the user wants, he can continue to tag the partially tagged text later.

The top portion of the window shows the text and the bottom part shows the morphological parses of the selected word in the top portion. The parses that are dropped by the morphological disambiguator are colored in red (e.g., the 2nd parse in Figure 4.2). If only one parse is left at the end of the disambiguation,


Supervised Part-of-Speech Tagger (04.05.2007)
File View Token Analysis Tools Help
Document:
< Previous
Next >
Choices:
Analysis
Log

Bayram günlerinde başrolü hep 'nostalji' kapar. Bu yıl da muhtemelen aynısı olacak ve bayram nostaljik hikayeler eşliğinde geçecek. Nostaljik hikayelere bir katkı da Pera Müzesi'nden geldi bu yıl. 20 Ekim'de açılan ve 7 Ocak'a kadar sürecek 'Konstantiniyye'den İstanbul'a XIX. Yüzyıl Ortalarından XX. Yüzyıla Boğaziçi'nin Rumeli Yakası Fotoğrafları' adlı sergi adından da anlaşılacağı gibi izleyenleri tarih tünelinde bir yolculuğa çıkarıyor. Suna ve İnan Kıraç'ın Eski İstanbul Fotoğrafları Koleksiyonu'ndan derlenen sergide 100'ü aşkın fotoğraf yer alıyor. Caranza, Robertson, Kargopoulo, Ferrier, Pascal Sebah, Sebah&Joaillier, Abdullah Kardeşler ve Gülmez Kardeşler gibi dönemin önemli fotoğrafçılarının imzasını taşıyan fotoğraflar hem Osmanlı başkentinin fiziksel ve sosyo-kültürel yapısı üstüne fikir veriyor hem de bugün çoğu şekil değiştirmiş veya kaybolmuş güzellikleri hatırlatıyor. Örneğin henüz köprülerin ve binaların arasında kaybolmamış olan Galata Rıhtımı. Ya da bugün yerinde yeller esen Tophane Meydanı. Bugün kültür merkezi ve sinema olarak hizmet veren, o yıllar Ortaköy sahilinde tek başına Boğaziçi'ni seyreden Feriye Karakolu...

[✓ 1- karakol+Noun+A3sg+P3sg+Nom]
2- karakol+Noun+A3sg+Pnon+Acc

Figure 4.2: Morphological Annotation Tool Operating on a Newspaper Article (Accessible at http://www.radikal.com.tr/haber.php?haberno=202413).

i.e., the token is fully disambiguated, that parse is automatically selected by the disambiguator. A human annotator can accept the selected parse by the morphological disambiguator, or he can select another parse by just clicking the parse that he thinks that it is the correct one.

4.4 Evaluation

4.4.1 Evaluation Method

In order to evaluate the performance of our morphological disambiguator, we created a testing data set that consists of 15 randomly selected Turkish articles from online newspapers. First, the selected articles are manually tagged so that the results of the morphological disambiguator can be compared with these manually tagged articles in order to evaluate its results. Initially there were 2454 tokens in the testing data set. The human expert detected 77 collocations in the testing data set, and there were 2370 tokens (single or composite) after all collocations are manually tagged. 329 of these 2370 tokens are punctuation tokens, and 2041 of them were non-punctuation tokens. Each of 2370 tokens is correctly tagged with a single correct parse by the human expert. The human expert also selected a correct parse for the tokens that are unhandled by the morphological analyzer (unknown tokens).

Each token is assigned a set of morphological parses by the morphological disambiguator. We expect that one of these parses to be the correct one. A token is fully disam-biguated if the disambiguator has dropped all parses except the correct one. We call the token correctly disambiguated if its multiple parses contain its correct parse.

We used the common precision and recall metrics in or-der to evaluate out morphological disambiguator. Precision measures the ratio of appropriate parses received from the morphological disambiguator to the total number of parses, while the recall measures the ratio of correctly disambigu-ated tokens to the total number of tokens. Precision and recall are calculated as given below:

$$\text{Precision} = \frac{\text{Number of correct parses in the result set}}{\text{Total number of parses in the result set}}$$

$$\text{Recall} = \frac{\text{Number of correctly disambiguated tokens in the result set}}{\text{Total number of tokens in the result set}}$$

4.4.2 Evaluation Results

After applying the morphological analyzer and the unknown token recognizer steps of the disambiguator, there were 2454 tokens and there were 4383 parses for those tokens. The distribution of the tokens into the number of parses can be seen in Table 4.5.

Then, the collocation recognizer is executed, and it results are given in Table 4.6. The collocation recognizer correctly found all of the 77 collocations. So, we can say that our collocation recognizer worked with 100% accuracy for this set. Although it worked with 100% accuracy for this set, some collocations can be missed in a larger testing data set. Our collocation recognizer is not be complete, however its coverage is very high. According to the results given in Table 4.6, the parses of each token contain its correct parse (100% recall), and 56.1% of all the parses in the result set are correct (56.1% precision). The results in Table 4.6 also indicate that the average number of parses per token is 1.78 (=2370/4226), and a token can have a maximum of 12 parses. These values were measured before the disambiguation process.

We calculated the precision and recall levels after applying the choose and delete rules (see Tables 4.7 and 4.8). The precision increases from 56.1% to 71.2% by applying the choose rules with only a small sacrifice (1.3%) from recall. The average number of parses per token also drops to 1.39 after the application of the choose rules.

Finally, we apply the delete rules in order to drop rare parses of certain tokens. By doing that we achieve a precision of 81.2% and the recall becomes 98.5%. The average number of parses per token also drops to 1.21. This is the overall performance of our morphological disambiguator. As a result, our disambiguator reduces the level of ambiguity from 1.78 parses per token to 1.21 parses per token with 81.2% precision and 98.5% recall values. In general, precision and recall are inversely proportional to each other, i.e., it is usual to sacrifice from recall in order to improve precision. As it can be seen from the results, the decrease in recall is small when compared to the much significant increase in the precision.

Table 4.5: The Results After the Morphological Analysis and Unknown Token Recognition.

# of parses	# of tokens
1	1340
2	701
3	190
4	157
5	29
6	16
7	1
8	10
9	1
10	1
11	0
12	8

Total Number of Tokens	2454
Total Number of Parses	4383

Table 4.6: The Results After Running the Collocation Recognizer.

# of parses	# of tokens	# of correctly disambiguated tokens
1	1304	1304
2	674	674
3	172	172
4	155	155
5	28	28
6	16	16
7	1	1
8	10	10
9	1	1
10	1	1
11	0	0
12	8	8

Number of Collocations	77
Total Number of Tokens	2370
Total Number of Parses	4226
Number of correctly disambiguated tokens	2370
Precision	**56.1%**
Recall	**100%**

Table 4.7: The Results After Applying Choose Rules.

# of parses	# of tokens	# of correctly disambiguated tokens
1	1820	1796
2	382	380
3	70	67
4	72	71
5	7	6
6	5	5
7	1	1
8	6	6
9	1	1
10	0	0
11	0	0
12	6	6

Total Number of Tokens	2370
Total Number of Parses	3283
Number of correctly disambiguated tokens	2339
Precision	**71.2%**
Recall	**98.7%**

Table 4.8: The Results After Applying Delete Rules.

# of parses	# of tokens	# of correctly disambiguated tokens
1	2010	1984
2	271	266
3	56	53
4	22	21
5	3	2
6	7	7
7	0	0
8	1	1
9	0	0
10	0	0
11	0	0
12	0	0

Total Number of Tokens	2370
Total Number of Parses	2873
Number of correctly disambiguated tokens	2334
Precision	**81.2%**
Recall	**98.5%**

Chapter 5

Learning From User Feedback

The availability of multiple possible template combinations that can be used to translate a given phrase and the ambiguities faced when converting the input phrase from the surface to the lexical form, results in multiple translation results for a given input. In order to present the user the most reliable results before the less reliable ones, we take advantage of the confidence factor assignment approach as described in Section 2.5.

The confidence factors are calculated merely from the translation examples in the learning phase. A problem with this scheme of confidence factor assignment is, that it does not consider the co-occurrence of the translation templates. Certain templates may be assigned low confidence factors when considered individually, but their co-existence in a translation result may require a different treatment, as the combination deserves a higher confidence. The reverse can also be true. In [29], Öz and Çiçekli proposed a modification to the original scheme of confidence factor assignment, that takes template combinations into consideration. This method calculates the confidence factors for template combinations in the learning phase, and once this is done the factors are never updated.

Moreover, the confidence factors learned from the translation examples in the learning phase may not always overlap with user expectations. Translation results correct for a given context, may be inappropriate for another context. A human translator can translate a phrase differently depending on the characteristics of the context of the text. Besides, different users may perceive the same translation result differently, depending on their background.

We could have encouraged the user, to add more translation examples in order to teach his preferences to the system. By adding enough number of new translation examples, the user can achieve to adjust the system to give the results, that best match his expectations, at the top. The disadvantage of this approach lies in its complexity. An ordinary user would not be able to estimate the number of new examples to add, in order to fine-tune the confidence factors assigned to the templates.

Instead, we propose a different mechanism for incorporating useful *user feedback* into the translation result ordering mechanism, which is one of the new features of our translation system. After each translation, the user has the option of evaluating the translation results in terms of their correctness. The system, using the information gathered by user interactions, ensures that the results marked in the evaluation as correct will be ranked above the results that are marked as incorrect, during the next translation of the same phrase.

The user interface provides two methods for inputting user feedback. The first one, *Shallow Evaluation*, lists the seach results in their bare surface-level representations; and the user can either mark a result as correct or incorrect. The second type of analysis is the *Deep Evaluation*, which is targeted for advanced users and provides the option of evaluating individual nodes of the parse trees built for each translation result. After inputting user feedback, the system learns context-dependent co-occurrence rules from that information.

5.1 Context-Dependent Co-occurrence Rules

In the previous versions of our system, only the confidence factors associated with translation templates were used for sorting the translation results. This method is not flexible, as the confidence factors are calculated in the learning phase and not updated throughout the system lifetime. Therefore, we propose the use of context-dependent co-occurrence rules in order to incorporate the user preferences into the result ordering mechanism. In our system, context-dependent co-occurrence rules are learned from user feedback in the translation phase, and continually updated throughout the lifetime of the system.

A context-dependent co-occurrence rule specifies a tree arrangement of translation templates and a list of contexts, each associated with a seperate aggregate confidence factor. For example, the rule

$$1(2, 3(5, 6), 4(7)) - [8_{(2)}, 9_{(4)}](0.7) \tag{5.1}$$

specifies the template tree $1(2, 3(5, 6), 4(7))$ and it has a single context $[8_{(2)}, 9_{(4)}]$, which is associated with the aggregate confidence value of 0.7. The template tree of this rule is depicted graphically in Figure 5.1. Here the numbers on the tree nodes denote the unique identifiers associated with each translation template.

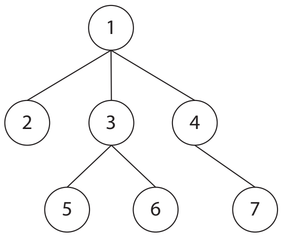


Figure 5.1: The Tree of Translation Templates of Rule (5.1).

In general, the tree structure used in co-occurrence rules are specified by the following context-free grammar (CFG):

$$\begin{aligned} T &\rightarrow template\_id \mid template\_id(ChildList) \\ ChildList &\rightarrow T \mid ChildList,\ T \end{aligned} \tag{5.2}$$

where T is the start symbol of the grammar. The ordering of the childen of a given node is not negligible, e.g., two trees, 1(2, 3) and 1(3, 2) are not equivalent.

A single template tree can be associated with several contexts, all of which having a seperate aggregate confidence factor. An examplary context-dependent co-occurrence rule is

$$1(2,\ 3) - [4_{(1)},\ 5_{(3)}](0.7) - [6_{(1)},\ 7_{(4)},\ 8_{(2)}](0.9), \tag{5.3}$$

in which a tree of translation templates, $1(2,\ 3)$, is followed by two contexts, $[4_{(1)},\ 5_{(3)}]$ and $[6_{(1)},\ 7_{(4)},\ 8_{(2)}]$, associated with aggregate confidence factors 0.7 and 0.9, respectively. The rule 5.3 is depicted graphically in Figure 5.2.

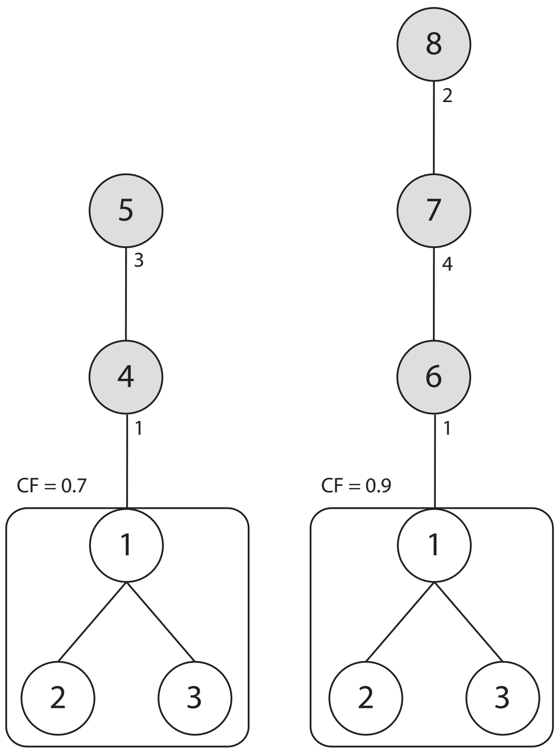


Figure 5.2: The Context-Dependent Co-occurrence Rule (5.3).

A context such as, $[4_{(1)},\ 5_{(3)}]$, specifies a sequence of translation templates, where each template is a child of the next template. In addition to that, each parent is marked with a subscript denoting the position of the child in the parent's list of children. For example, for the context $[4_{(1)},\ 5_{(3)}]$, the tree $1(2,\ 3)$ is the 1^{st} child of template 4; and template 4 is the 3^{rd} child of template 5.

As we aim bidirectinal translation, two sets of co-occurrence rules are maintained in the system, one of which is used in English to Turkish translations and the other in the reverse direction. As the user runs the translation component and evaluates the generated translation results, the co-occurrence rules are continually updated, i.e., new rules are learned and context information of existing rules are updated.

5.1.1 Using the Context-Dependent Co-occurrence Rules

A co-occurrence rule specifies an aggregate confidence factor. If the parse tree built for a translation result has a subtree matching the rule, then this aggregate confidence factor overrides the individual confidence factors in that subtree. For example, assume that during the translation of the English phrase

$$\text{“red haired man”} \tag{5.4}$$

the translation templates given below are used:

$$\begin{aligned}
1 &: X^1_{Adj\ Noun\ Sg}\ \hat{}\text{DB+Adj+Ed}\ X^2_{Noun}\ \text{+Sg} \leftrightarrow \\
&\quad Y^1_{Adj\ Noun\ A3sg\ Pnon\ Nom}\ \hat{}\text{DB+Adj+With}\ Y^2_{Noun}\ \text{+A3sg +Pnon +Nom} \\
2 &: X^1_{Adj}\ X^2_{Noun\ Sg} \leftrightarrow Y^1_{Adj}\ Y^2_{Noun\ A3sg\ Pnon\ Nom} \\
3 &: \text{man+Noun} \leftrightarrow \text{adam+Noun} \\
4 &: \text{red+Adj} \leftrightarrow \text{kızıl+Adj} \\
5 &: \text{hair+Noun +Sg} \leftrightarrow \text{saç+Noun +A3sg +Pnon +Nom}
\end{aligned} \tag{5.5}$$

where the English to Turkish confidence factors of individual templates are 0.8, 0.7, 1.0, 0.5, and 1.0 respectively. Suppose that the parse tree in Figure 5.3 is built during the generation of a translation result,

$$\text{“kızıl saçlı adam”.} \tag{5.6}$$

Therefore, the confidence value of this translation is

$$confidence = 0.8 \times 0.7 \times 1.0 \times 0.5 \times 1.0 = 0.28. \tag{5.7}$$

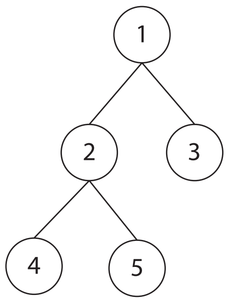


Figure 5.3: Parse Tree Built for the Translation of Phrase 5.4.

Now, suppose that a co-occurrence rule that specifies an aggregate confidence factor for the partial translation “red haired → kızıl saçlı”, such as

$$2(4,5) - [1_{(1)}](0.9), \tag{5.8}$$

is learned beforehand. Then, as the template tree specified in the co-occurrence rule, matches the subtree 2(4,5) in the parse tree of the result and the context of the matching subtree is $[1_{(1)}]$, the aggregate confidence factor specified in the rule overrides the original confidence factors of the nodes in the matching subtree; and the new confidence value of the translation result becomes

$$confidence = 0.8 \times 0.9 \times 1.0 = 0.72. \tag{5.9}$$

Now we can formalize the confidence value calculation method exemplified above by giving Algorithm 4. Running this algorithm with the parameter *node* set to the root of the parse tree in Figure 5.3 will also return the confidence value of 0.72 as the result.

Confidence-Value-Exact defines the confidence value of a parse tree recursively. If at any point of recursion, a rule matching the subtree rooted at the current parse tree node can be found, and a context matching the context of the current parse tree node is available in the rule, then the associated aggregate confidence value is returned. If these conditions are not satisfied, then the values returned by Confidence-Value-Exact(*child*), for all *child* in the children set of *node*, are multiplied with the confidence factor of the template represented by

```
CONFIDENCE-VALUE-EXACT($node$)
 1: $tree \leftarrow$ the tree rooted at $node$
 2: $context \leftarrow$ the context of $node$
 3: if there exists a co-occurrence rule $R$ that matches $tree$ and
      there exists a context, $R\_context$, in $R$, where $R\_context = context$ then
 4:     return the aggregate confidence factor associated with $R\_context$
 5: else
 6:     $confidence \leftarrow$ confidence factor of the template represented by $node$
 7:     $children \leftarrow \{child : child$ is a child of $node\}$
 8:     for all $child \in children$ do
 9:         $confidence \leftarrow confidence \times$ CONFIDENCE-VALUE-EXACT($child$)
10:     end for
11:     return $confidence$
12: end if
```

Algorithm 4: CONFIDENCE-VALUE-EXACT. Returns the confidence value of a translation result.

node; and the result of this multiplication is retured.

5.1.2 The Concept of User Profiles

The context-Dependent co-occurrence rules learned from user feedback reflect the preferences of a particular user. Translation characteristics vary from one human translator to another, and usually there are numerous correct translations of a given text. Therefore, we use the concept of user profiles in our system.

When a user evaluates the results of a translation, the co-occurrence rules learned from the evaluation are kept in his own user profile. Thus, other user profiles in the system are not affected. Also, a single user can create multiple profiles, each of which is used for a different text context — such as science, literature, law, etc. — that has a distinct characteristic.

5.2 Learning Context-Dependent Co-occurrence Rules

In our system, the context-dependent co-occurrence rules are learned from user feedback. After retrieving the translation results, a user has the option of evaluating them. As stated earlier, the system provides two different evaluation interfaces, the *Shallow Evaluation*, which provides minimum detail for inexperienced users, and the *Deep Evaluation*, which is targeted for advanced users.

5.2.1 Deep Evaluation of Translation Results

The Deep Evaluation is targeted for advanced users and can be used to learn more fine-tuned co-occurrence rules compared to Shallow Evaluation. In the Deep Evaluation, the user can evaluate individual nodes of the parse tree associated with each translation result.

The user interface provides two check boxes for each node of a parse tree in order to input the correctness judgement from the user. The first check box, Check Box 1, can be set to 3 different values, which are *correct* (☑), *incorrect* (☒) and *indeterminate* (☐). The indeterminate state can be chosen for a node when the user does not want to evaluate the subtree rooted at that one. Check Box 1 is always shown to the user, whereas the second check box, Check Box 2, is only shown when Check Box 1 is set to incorrect and the node also has a child evaluated as incorrect. Check Box 2 has two states, namely *correct* (☑) and *incorrect* (☒). The different configurations of the two check boxes constitute a total of 5 states for the nodes, the meanings of which are explained in detail in Table 5.1.

For a given node, the user determines the state of Check Box 1 by answering the question: "***Is the translation implied by the subtree rooted at this node correct?***". Therefore, if Check Box 1 is set to (☑) for a node, then the partial translation implied by the subtree rooted at that node must be correct.

Table 5.1: States Used in Deep Evaluation.

State	**Symbol**	**Explanation**
1	☐	This is the initial state assigned to every node at the beginning of the evaluation. It simple denotes, that no node exists in the subtree rooted at this node that is evaluated by the user.
2	☑	This state denotes, that the user evaluated the partial translation, which is implied by the nodes in the subtree rooted at this node, as *correct*. It also indicates, that all children of this node are also in state 2.
3	☒	This state denotes, that the user evaluated the partial translation, which is implied by the subtree rooted at this node, as *incorrect*. It also indicates, that the user has not evaluated any of the children nodes as incorrect, or the node is a leaf.
4	☒☒	This state denotes, that the user evaluated the partial translation, which is implied by the subtree rooted at this node, as *incorrect*. The difference from state 3 is, that, in order for a node to be in this state, the node has to have a child that is evaluated as incorrect.
5	☒☑	This state has all properties of state 4. In addition to that, this state denotes that, although the translation is erroneous, the use of this translation template in the current context is not the cause of the error. That is, the translation error is isolated in some children of this node.

Likewise, if it is set to (☒) then the partial translation implied by the subtree rooted at that node is incorrect.

Similarly, for a given node, the user determines the state of Check Box 2 by answering the question: "***Can the translation error be isolated to some erroneous child(s) of this node?***". If the partial translation implied by the subtree rooted at a node is incorrect, that node may not be the actual source of the translation error. In other words, the error can be isolated at one or more children nodes. If this is the case, the Check Box 2 is set to (☑) denoting that the node is not a cause for the erroneous translation. If the error cannot be isolated to a child node, then Check Box 2 is set to (☒).

As an example, suppose that the translation system knows only the following translation templates:

$$\begin{aligned}
&1: X^1_{Adj\ Noun\ Sg\ \hat{}DB+Adj+Ed}\ X^2_{Noun}\ \text{+Sg} \leftrightarrow \\
&\quad Y^1_{Adj\ Noun\ A3sg\ Pnon\ Nom\ \hat{}DB+Adj+With}\ Y^2_{Noun}\ \text{+A3sg +Pnon +Nom} \\
&2: X^1_{Adj}\ X^2_{Noun\ Sg}\ \text{\^{}DB+Adj+Ed} \leftrightarrow Y^1_{Adj}\ Y^2_{Noun\ A3sg\ Pnon\ Nom}\ \text{\^{}DB+Adj+With} \\
&3: \text{blond+Adj}\ X^1_{Noun}\ \text{+Sg} \leftrightarrow \\
&\quad \text{sarı+Adj saç+Noun +A3sg +Pnon +Nom \^{}DB+Adj+With}\ Y_{Noun}\ \text{+A3sg +Pnon +Nom} \\
&4: \text{hair+Noun +Sg} \leftrightarrow \text{saç+Noun +A3sg +Pnon +Nom} \\
&5: \text{woman+Noun} \leftrightarrow \text{kadın+Noun} \\
&6: \text{yellow+Adj} \leftrightarrow \text{sarı+Adj}
\end{aligned} \tag{5.10}$$

where the Turkish to English confidence factors are 0.9, 0.8, 0.5, 1.0, 1.0 and 1.0, respectively. Assume that the user has translated the Turkish phrase

$$\text{“sarı saçlı kadın”,} \tag{5.11}$$

and the translation system returned two different results, as shown in Figure 5.4.

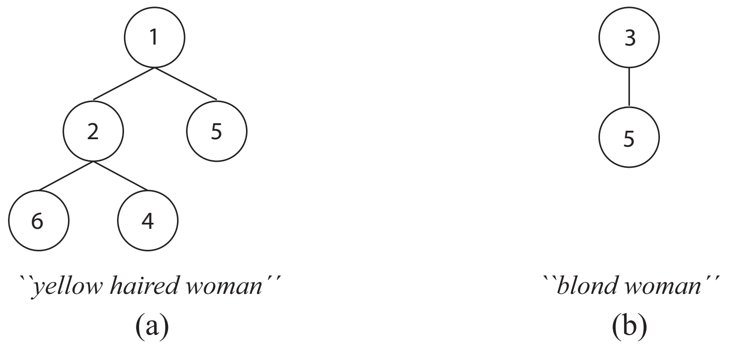


Figure 5.4: Translation Results for Examplary Phrase 5.11.

The first result, “*yellow haired woman*” is a literal translation and it is less appropriate when compared to the second one, “*blond woman*”. However the confidence value of the first translation, $0.9 \times 0.8 \times 1.0 \times 1.0 \times 1.0 = 0.72$, is greater than the confidence value, $0.5 \times 1.0 = 0.5$, of the second one; therefore, the first translation is listed over the second one.

However, suppose that the user prefers the second translation, "*blond woman*", over the first one. In that case, the user may enter the Deep Evaluation screen to teach his preference to the system.

To simplify the evaluation process, rather than showing the contents of the non-atomic templates as node labels, the Deep Evaluation screen shows the partial translations implied by those node. The partial translation implied by a node is defined recursively, and found by replacing each variable in the template by the partial translation implied by the corresponding child node. Since the leaf nodes always represent an atomic template in the parse tree of a result, the partial translation implied by a node can always be found. Also the partial translation of the root node is equal to the lexical form of the translation result associated with that tree. For example, for the template tree of the first result in our example, rather than showing the contents of the 2^{nd} template as the label of the node 2, the Deep Evaluation screen shows the partial translation implied by node 2, which is

$$\begin{gathered}\text{sarı+Adj saç+Noun +A3sg +Pnon +Nom ˆDB+Adj+With} \rightarrow \\ \text{yellow+Adj hair+Noun +Sg ˆDB+Adj+Ed}\end{gathered} \tag{5.12}$$

At the beginning of the evaluation, in order to simplify the user interface, the roots of the translation trees are collapsed, i.e., the children of the root nodes are hidden from the user. The children of a node are only expanded (shown) when the partial translation implied by that node is evaluated as incorrect by the user. By using this method, the user marks paths from the root to the subtrees that are the sources of erroneous translation.

For translation results that are perceived as correct by the user, the evaluation is simple. When the user marks the root node of the parse tree of such a translation result as correct, all other nodes in the parse tree are considered to be correct as well. This is intuitive, as we expect a correct translation to be made up of partial translations, that are correct in the context of the translated phrase. In

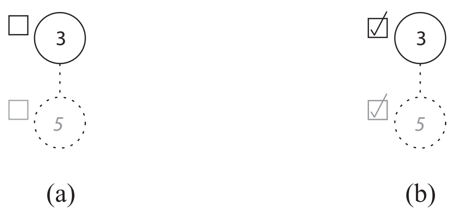


Figure 5.5: Evaluation of the Translation Result Given in Figure 5.4(b).

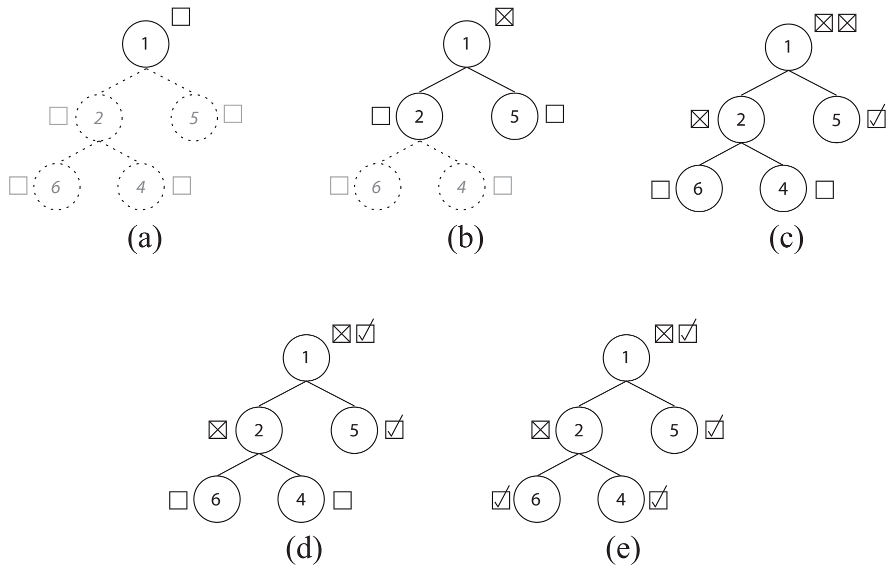


Figure 5.6: Evaluation of the Translation Result Given in Figure 5.4(a).

our example, the user perceives the second translation result, "*blond woman*", as correct. So, the node 5 in the parse tree of that result will be marked as correct along with the root node. The Deep Evaluation process for this result is depicted in Figures 5.5(a–b).

For the translation results that are perceived as incorrect, or *inappropriate*, by the user, the evaluation requires more attention. The user starts by setting the root node to state ☒, and walks through the tree by expanding the nodes on the paths to erroneous subtrees. For our examplary translation result "*yellow haired woman*", the process of Deep Evaluation is depicted in Figures 5.6(a–e). One should note that, although this translation result is not a completely wrong one, it is less desired compared to the other result. To teach his preference to the system, a user can treat such a result as if it was incorrect[1].

Initially, only the root node is shown to the user (Figure 5.6(a)), along with the translation result in its lexical form. When the user sets the state of the root node to ⊠, the root node is expanded and its children are shown (Figure 5.6(b)).

As the partial translation implied by node 5, "kadın+Noun → woman+Noun", is correct, the user sets the state of node 5 as ☑. Since the partial translation implied by node 2, as given in 5.12, is perceived as incorrect, the user sets the state of node 2 to ⊠ (Figure 5.6(c)). Also, as the error can be isolated in node 2, the user changes the state of the root node to ⊠☑ (Figure 5.6(d)).

Lastly, the user evaluates the nodes 6 and 4. Node 6 implies the partial translation "sarı+Adj → yellow+Adj". Using this node in the context of $[2_{(1)}, 1_{(1)}]$ is not wrong. Similarly, node 4 could well be used in the same context correctly if node 6 was not there. In other words, the cause of the error is using nodes 6 and 4 together. When considered separately, using these nodes in the context they appear is not wrong. So, in the Deep Evaluation, the states of both of the nodes are set to ☑ by the user (Figure 5.6(e)).

In order to undestand the Deep Evaluation of the translation results given above better, see Appendix A, which provides images of the actual Deep Evaluation GUI.

5.2.2 Determining The Desired Confidence Values

Regardless of the evaluation method used, each translation result is either marked as correct or incorrect. The user also has the option of leaving a translation result unevaluated. In that case, no co-occurrence rule is learned from that particular translation result. As stated earlier, we use the symbols □, ⊠ and ☑, to denote

[1]In Deep Evaluation, treating a not-that-appropriate result as if it was incorrect is safe, as the system will never assign a 0 confidence factor to such a translation result. Learned co-occurrence rules will be fine-tuned to place this kind of results just below the more desired ones.

Table 5.2: Sample Translation Result Evaluation.

Translation Result	Original Confidence Value	Evaluation Assessment
A	0.9	☒
B	0.8	☐
C	0.6	☑
D	0.4	☑
E	0.3	☒

unevaluated, incorrect and correct translation results, respectively.

The co-occurrence rules learned from a user evaluation guarantee, that during the next translation of the same input phrase, results marked as correct will be placed above results marked as incorrect, i.e., learned rules will adjust the confidence value of correct and incorrect translations in such a way, that confidence values of correct translations will be higher than that of incorrect translations.

Suppose that the translation of an input phrase returned 5 different results, A, B, C, D and E; and the user evaluated the results as shown in Table 5.2. From Table 5.2, we can see that the translation results except B were evaluated. While A is the result with the highest confidence value, it is marked as incorrect. Although, C and D are marked as correct, they are assigned lower confidence values compared to A, thus ranking below A. Therefore, the co-occurrence rules, that will be learned from the evaluation should change the order of A, C and D in such a way, that A comes below C and D. Even though E is marked as incorrect, we do not have to change its position in the ordering, since there are no correct results with confidence values lower than that of E. So, we will not learn any rules from E.

The next step for learning co-occurrence rules, is to determine desired confidence values for the translation results. In order to do that, we have to calculate 6 values, namely $lower\_hinge$, $upper\_hinge$, $length_1$, $length_2$, gap_{avg} and $scale\_factor$. The first four of these values for the example in Table 5.2 are shown in Figure 5.7.

Let the incorrect translation result with the highest confidence value be

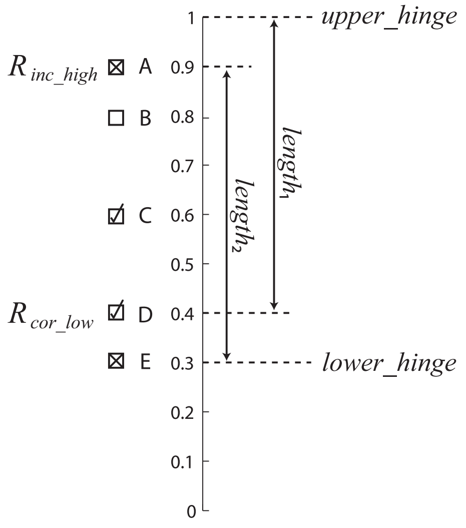


Figure 5.7: *lower_hinge*, *upper_hinge*, $length_1$ and $length_2$ for the Example in Table 5.2.

R_{inc_high} and the correct result with the lowest confidence value be R_{cor_low}. *Upper_hinge* is the confidence value of the correct result that is ranked just above R_{inc_high}. If such a correct result does not exits, then $upper\_hinge = 1$. Symmetrically, *Lower_hinge* is the confidence value of the incorrect result that is ranked just below R_{cor_low}. If such an incorrect result does not exist, then $lower\_hinge = 0$. Also, $length_1$ and $length_2$ are defined as

$$length_1 = |upper\_hinge - confidenceOf(R_{cor\_low})|, \tag{5.13}$$

$$length_2 = |lower\_hinge - confidenceOf(R_{inc\_high})|. \tag{5.14}$$

The average gap, gap_{avg}, between the original confidence values of subsequent evaluated translation results in range [*lower_hinge*, *upper_hinge*] for Table 5.2 is

$$gap_{avg} = \frac{(0.9 - 0.6) + (0.6 - 0.4) + (0.4 - 0.3)}{3} = 0.2. \tag{5.15}$$

Lastly, the *scale_factor* is calculated as

$$scale\_factor = \frac{upper\_hinge - lower\_hinge}{length_1 + gap_{avg} + length_2}, \tag{5.16}$$

which is $(1-0.3)/(0.6+0.2+0.6) = 0.7/1.4 = 0.5$ for our examplary evaluation.

After calculating the *scale_factor*, the desired confidence value of a translation result R, that is in range (*lower_hinge*, *upper_hinge*) is assigned by Formula 5.17.

$$\begin{array}{l}\text{desired}\\ \text{confidence of } R\end{array} = \begin{cases} confidenceOf(R) & \text{if } R \text{ is not evaluated,} \\ \\ upper\_hinge - (upper\_hinge - \\ \quad confidenceOf(R)) \times scale\_factor & \text{if } R \text{ is correct,} \\ \\ lower\_hinge + (confidenceOf(R) - \\ \quad lower\_hinge) \times scale\_factor & \text{if } R \text{ is incorrect.} \end{cases} \tag{5.17}$$

For our examplary evaluation, the correct results have been ranked above the incorrect ones after assigning the desired confidence values, as shown in Table 5.3. The process is depicted graphically in Figure 5.8.

Table 5.3: The New Ranking of the Results in Table 5.2.

Translation Result	Desired Confidence Value	Evaluation Assessment
B	0.8	☐
C	0.8	☑
D	0.7	☑
A	0.6	☒
E	0.3	☒

One should note, that our formula in 5.17 preserves the order among correct results, which is also true for incorrect results.

Now, let us return back to our example translation of the Turkish phrase (5.11):

"sarı saçlı kadın".

In our examplary scenario, the translation system had returned two different

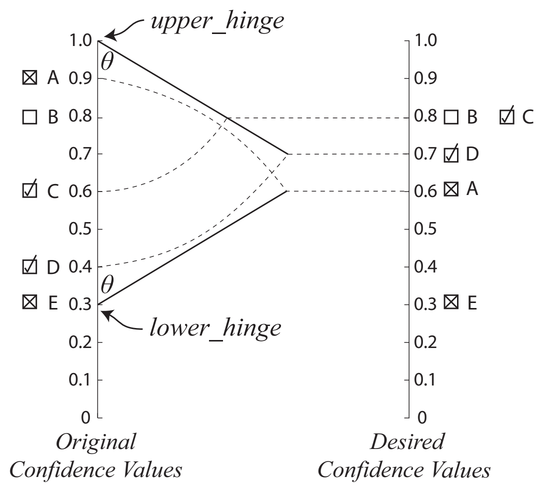


Figure 5.8: Assigning the Desired Confidence Values. ($\theta = arccos(scale\_factor)$)

translation results for this input phrase, which are shown below with the coresponding confidence values:

$$\text{“yellow haired woman”}: 0.72,$$
$$\text{“blond woman”}: 0.5.$$

In this case, when we apply the methods described in this section we will obtain the following parameters:

$$\begin{aligned} lower\_hinge &= 0.0 \\ upper\_hinge &= 1.0 \\ length_1 &= 0.5 \\ length_2 &= 0.72 \\ gap_{avg} &= 0.22 \end{aligned}$$

Therefore,

$$\begin{aligned} scale\_factor &= \frac{upper\_hinge - lower\_hinge}{length_1 + gap_{avg} + length_2} \\ &= \frac{1}{0.5 + 0.22 + 0.72} \\ &= 0.694. \end{aligned}$$

Using Formula (5.17), the desired confidence values for the translation results become:

$$\begin{aligned} \text{“yellow haired woman”}: &\ 0.499, \\ \text{“blond woman”}: &\ 0.653. \end{aligned} \tag{5.18}$$

One should note, that the desired confidence values comply with the expectations of the user. The more proper result, “*blond woman*”, has a higher desired confidence value then that of the first result, “*yellow haired woman*”.

The last step in learning context-dependent co-occurrence rules consists of their extraction from the parse trees using the desired confidence values calculated as described above. The next subsection shows how to realize this.

5.2.3 Extracting Context-Dependent Co-occurrence Rules

The last step in learning co-occurrence rules, is to extract them from the parse trees of the evaluated translation results. After finding the desired confidence values for each translation result in range (*lower_hinge*, *upper_hinge*), the system extracts context-dependent co-occurrence rules from those results, using the Extract-Rules procedure given in Algorithm 5.

The first parameter to this procedure is an array of translation results, while the second parameter is an array of desired confidence values. The desired confidence value for each translation result is calculated as described in Section 5.2.2. Extract-Rules uses Extract-Rules-Incorrect and Extract-Rules-Correct procedures (given in Algorithms 6 and 7, respectively), as subroutines.

Extract-Rules-Incorrect procedure is used to extract co-occurrence rules from the parse trees of translation results that are marked as incorrect by the user. Extract-Rules-Incorrect performs a depth-first traversal on the parse tree of a given incorrect result. During the traversal, subtrees rooted at nodes marked as □ or ⊠☑ are not explored, since we want to learn rules from subtrees

```
EXTRACT-RULES($Results$, $new\_confidences$)
 1: if $length[Results] \neq length[new\_confidences]$ then
 2:    Error: lengths of the two arrays must match
 3: else
 4:    for $i = 1$ to $length[Results]$ do
 5:       $root \leftarrow$ root node of $Results[i]$
 6:       $context \leftarrow [\,]$ //an empty context.
 7:       if $root$ is in state ☑ then
 8:          EXTRACT-RULES-CORRECT($root$, $context$, $new\_confidences[i]$)
 9:       else if $root$ is in state ☒, ☒☒or ☒☑ then
10:          EXTRACT-RULES-INCORRECT($root$, $context$, $new\_confidences[i]$)
11:       end if
12:    end for
13: end if
```

Algorithm 5: EXTRACT-RULES. Extracts context-dependent co-occurrence rules from evaluated translation results.

that cause the incorrect translation. Therefore, EXTRACT-RULES-INCORRECT learns context-dependent co-occurrence rules only from subtrees rooted at nodes that are marked as ☒ and ☒☒.

The children of a node marked as ☒ will never be explored during the depth-first traversal, since such a node cannot have an incorrect children. On the other hand, the childen of nodes marked as ☒☒ or ☒☑ will be explored, as this kind of a node must have at least one incorrect children.

EXTRACT-RULES-INCORRECT procedure takes 3 arguments. The first one is a node in the parse tree of a translation result, the second one is the context in which the node exists. The last argument is the desired confidence for the subtree rooted at the given node. This procedure works as follows: Assume that a node p has children $c_1,\ c_2,\ \ldots,\ c_n$, where $c_1,\ c_2, \ldots,\ c_k$ are marked as incorrect (☒, ☒☒ or ☒☑) and $c_{k+1}, \ldots,\ c_n$ are either marked as correct (☑) or left unevaluated (□). When EXTRACT-RULES-INCORRECT is called for the node p, with the desired confidence value *desired-confidence*$_p$, first a context-dependent co-occurrence rule is learned, if appropriate. The learned rule will have an aggregate confidence factor that is lower than the original confidence value of the subtree rooted at p, penalizing the subtree.

```
EXTRACT-RULES-INCORRECT($node$, $context$, $desired\_confidence$)
 1: if state of the $node$ is ⊠, ⊠⊠ or ⊠☑ then
 2:    $tree \leftarrow$ the tree rooted at $node$
 3:    if state of $node$ is ⊠ or ⊠⊠ then
 4:       LEARN-RULE($tree, context, desired\_confidence$)
 5:    end if
 6:    $old\_confidence \leftarrow$ the confidence value of $tree$
 7:    $incorrect\_children \leftarrow \{c : c$ is a child of $node$ in ⊠, ⊠⊠ or ⊠☑ state$\}$
 8:    if $incorrect\_children \neq \emptyset$ then
 9:       $\beta \leftarrow (desired\_confidence/old\_confidence)^{1/|incorrect\_children|}$
10:       for each $child \in incorrect\_children$ do
11:          $index \leftarrow getChildIndex(node, child)$
12:          $child\_context \leftarrow add(copy(context), \langle node, index \rangle)$
13:          $child\_confidence \leftarrow$ confidence value of the subtree rooted at $child$
14:          EXTRACT-RULES-INCORRECT($child, child\_context, \beta \times child\_confidence$)
15:       end for
16:    end if
17: end if
```

Algorithm 6: EXTRACT-RULES-INCORRECT. Extracts context-dependent co-occurrence rules from incorrect translation results.

Then, an incorrect-child-multiplier value β is calculated as

$$\beta = \sqrt[k]{\frac{\textit{desired-confidence}_p}{\textit{original-confidence}_p}} \tag{5.19}$$

where *original-confidence*$_p$ is the original confidence value of the tree rooted at node p. This multiplier is used to distribute the penalty evenly to each of the incorrect child nodes of p. One should note that, the inequality

$$\textit{desired-confidence}_p / \textit{original-confidence}_p < 1$$

will always hold, as p is incorrect, therefore $\beta < 1$ is also true.

Next, for each child c_i, $1 \leq i \leq k$, Extract-Rules-Incorrect is called recursively with the desired confidence parameter

$$\textit{desired-confidence}_{c_i} = \beta \times \textit{original-confidence}_{c_i}.$$

Thus, for $1 \leq i \leq k$,

$$\textit{desired-confidence}_{c_i} < \textit{original-confidence}_{c_i}.$$

Extract-Rules-Correct is very similar to Extract-Rules-Incorrect, except it is used to learn rules from correct translations. As all nodes in the parse tree of a correct translation result would be marked as ☑, the depth-first traversal performed by recursive calls of Extract-Rules-Correct will effectively expore all the nodes in such a tree. This procedure works as follows: Assume that a node p has children $c_1, c_2, \ldots, c_m$, where all the children are marked as correct. When Extract-Rules-Correct is called for the node p, with the desired confidence value *desired-confidence*$_p$, first a context-dependent co-occurrence rule that rewards the subtree rooted at p is learned, if appropriate. Then a correct-child-multiplier value δ is calculated as

$$\delta = \sqrt[m]{\frac{\textit{desired-confidence}_p}{\textit{original-confidence}_p}} \qquad (5.20)$$

where *original-confidence*$_p$ is the original confidence value of the tree rooted at node p. This multiplier is used to distribute the reward evenly to each of the correct child nodes of p. One should note that, the inequality

$$\textit{desired-confidence}_p / \textit{original-confidence}_p > 1$$

will always hold, as p is correct, therefore $\delta > 1$ is also true.

```
EXTRACT-RULES-CORRECT($node$, $context$, $desired\_confidence$)
 1: if state of the $node$ is ☑ then
 2:    $tree \leftarrow$ the tree rooted at $node$
 3:    if $desired\_confidence \leq 1$ then
 4:       LEARN-RULE($tree, context, desired\_confidence$)
 5:    else
 6:       LEARN-RULE($tree, context, 1$)
 7:    end if
 8:    $old\_confidence \leftarrow$ the confidence value of $tree$
 9:    $correct\_children \leftarrow \{c : c$ is a child of $node$ in ☑ state$\}$
10:    if $correct\_children \neq \emptyset$ then
11:       $\delta \leftarrow (desired\_confidence/old\_confidence)^{1/|correct\_children|}$
12:       for each $child \in correct\_children$ do
13:          $index \leftarrow getChildIndex(node, child)$
14:          $child\_context \leftarrow add(copy(context), \langle node, index\rangle)$
15:          $child\_confidence \leftarrow$ confidence value of the subtree rooted at $child$
16:          EXTRACT-RULES-CORRECT($child, child\_context, \delta \times$
             $child\_confidence$)
17:       end for
18:    end if
19: end if
```

Algorithm 7: EXTRACT-RULES-CORRECT. Extracts context-dependent co-occurrence rules from correct translation results.

Next, for each child c_i, $1 \leq i \leq m$, EXTRACT-RULES-CORRECT is called recursively with the desired confidence parameter

$$\textit{desired-confidence}_{c_i} = \delta \times \textit{original-confidence}_{c_i}.$$

Thus, for $1 \leq i \leq m$,

$$\textit{desired-confidence}_{c_i} > \textit{original-confidence}_{c_i}.$$

Note that, for some child c_i, $1 \leq i \leq m$, the inequality $\textit{desired-confidence}_{c_i} > 1$ can be true, since $\delta > 1$. This is not allowable[2], as we do not want to learn a context-dependent co-occurrence rule with an aggregate confidence factor > 1.

[2]A confidence factor represents a probability value, therefore cannot be > 1.

Extract-Rules-Correct prevents this kind of situations by simply setting the rule confidence to 1 for subtrees rooted at such c_i.

Now, let us return back to our examplary Deep Evaluation scenario for the translation results of the Turkish phrase (5.11). The parse trees of the translation results were evaluated as shown in Figures 5.6(e) and 5.5(b), and the desired confidence values were determined as given in (5.18). In the last step, we will extract context-dependent co-occurrence rules from the parse trees of these translation results.

The first translation result was an incorrect one. Therefore, Extract-Rules will call Extract-Rules-Incorrect for the root node of the parse tree of this result, with the desired confidence value of 0.499. As the root node is marked as ⊠☑, no rules will be extracted at that node. Then the β value will be calculated as

$$\beta = \sqrt[1]{\frac{0.499}{0.72}} = 0.693. \tag{5.21}$$

Next, Extract-Rules-Incorrect will be called for the incorrect child of the root, which is node 2, recursively, with the desired confidence value parameter of $\beta \times 0.8 = 0.693 \times 0.8 = 0.554$, where 0.8 is the original confidence value of the subtree rooted at node 2. Since node 2 is marked as ☑ the context dependent co-occurence rule

$$2(6,\, 4) \; - \; [1_{(1)}](0.554) \tag{5.22}$$

will be extracted. Since there are no erroneous nodes in the tree, this rule will be the only rule that is learned for this translation result.

The second translation result was a correct one. Therefore, Extract-Rules will call Extract-Rules-Correct for the root node of the parse tree of this result, with the desired confidence value of 0.653. As this node is marked as ☑, the context dependent co-occurence rule

$$3(5) \; - \; [\;](0.653) \tag{5.23}$$

will be extracted. Note that the only context associated with this rule is an empty one, as the rule was extracted from the root node. Then, the δ value will be calculated as

$$\delta = \sqrt[1]{\frac{0.653}{0.5}} = 1.306. \tag{5.24}$$

Next, EXTRACT-RULES-CORRECT will be called for the correct child of the root, which is node 5, recursively, with the desired confidence value parameter of $\delta \times 1.0 = 1.306 \times 1.0 = 1.306$, where 1.0 is the original confidence value of the subtree rooted at node 5. Since the desired confidence value is greater than 1, the extracted rule will be assigned the maximum possible aggregate confidence factor, which is 1. Therefore the second extracted rule will be

$$5 \;-\; [3](1.0). \tag{5.25}$$

As no other nodes remain in the parse tree, the execution will be over.

5.2.4 Shallow Evaluation of Translation Results

Shallow Evaluation is the second evaluation interface of our translation system, which is targeted for inexperienced users, as it provides much simpler means of user interaction, compared to Deep Evaluation. In Shallow Evaluation, translation results are showed in their surface forms, instead of their lexical forms as in Deep Evaluation. This makes it much more easier to interpret the results during the evaluation.

While in Deep Evaluation, the nodes in the parse trees of translation results can be evaluated individually by the user, in Shallow Evaluation the user makes only a single correctness judgement for each result. Thus, a translation result is either marked as correct (☑) or incorrect (☒), or left unevaluated (☐).

Due to various morphological level ambiguities that exist in the source and target languages, two translation results with distinct lexical forms can map to the same surface form. In such cases, those results are presented to the user in a group and the correctness judgement of the user for that group is assigned to all group members.

In fact, Shallow Evaluation is a front-end to Deep Evaluation with a simplified interface. Shallow Evaluation input taken from the user is automatically converted to an instance of Deep Evaluation input, on which the co-occurrence rule learning methods described in the previous subsections are applied. IMITATE-DEEP-ANALYSIS procedure, given in Algorithm 8 performs this input conversion. An example run of this algorithm is given in Figure 5.9.

In IMITATE-DEEP-ANALYSIS, Each incorrect result is compared with the correct results. For comparison, COMPARE-TREES procedure is used as a subprocedure. During successive comparisons, the nodes that might have caused the incorrect translation is tried to be identified. Note that COMPARE-TREES ensures, that the comparison order does not change the final configuration of the incorrect results.

When an incorrect result is compared with the correct results, in some rare occasions, all nodes in the parse tree of that incorrect result may be set to ☑. This is an undesired effect, as it prevents learning any co-occurrence rules from that particular incorrect result. This happens if successive comparisons validate all of the nodes in the parse tree of an incorrect result. Lines 11–14 of IMITATE-DEEP-ANALYSIS handles this situation and sets the root node to an incorrect state. Therefore, it is guaranteed that at least one co-occurrence rule is extracted from each incorrect result.

```
IMITATE-DEEP-ANALYSIS(correct_results_list, incorrect_results_list)
 1: for each correct_result ∈ correct_results_list do
 2:     INITIALIZE-TREE(correct_result, ☑)
 3: end for
 4: for each incorrect_result ∈ incorrect_results_list do
 5:     INITIALIZE-TREE(incorrect_result, □)
 6:     incorrect_root ← the root node in the parse tree of incorrect_result
 7:     for each correct_result ∈ correct_results_list do
 8:         correct_root ← the root node in the parse tree of correct_result
 9:         COMPARE-TREES(incorrect_root, correct_root)
10:     end for
11:     if state of incorrect_root is ☑ then
12:         INITIALIZE-TREE(incorrect_result, □)
13:         set state of incorrect_root to ☒
14:     end if
15: end for
INITIALIZE-TREE(result, state)
 1: set all nodes in the parse tree of result to state
COMPARE-TREES(incorrect_root, correct_root)
 1: if state of incorrect_root is ☑ then
 2:     return true
 3: else if template no of incorrect_root = template no of correct_root then
 4:     flag ← true
 5:     for $i = 1$ to $n$, where $n$ is the number of children of incorrect_root do
 6:         incorrect_child ← $i^{th}$ child of incorrect_root
 7:         correct_child ← $i^{th}$ child of correct_root
 8:         if COMPARE-TREES(incorrect_child, correct_child) = false then
 9:             flag ← false
10:         end if
11:     end for
12:     if flag = true then
13:         set state of incorrect_root to ☑
14:     else
15:         set state of incorrect_root to ☒☒
16:     end if
17:     return flag
18: else
19:     if state of incorrect_root is □ then
20:         set state of incorrect_root to ☒
21:     end if
22:     return false
23: end if
```

Algorithm 8: IMITATE-DEEP-ANALYSIS. Converts a shallow evaluation input to a deep evaluation input automatically.

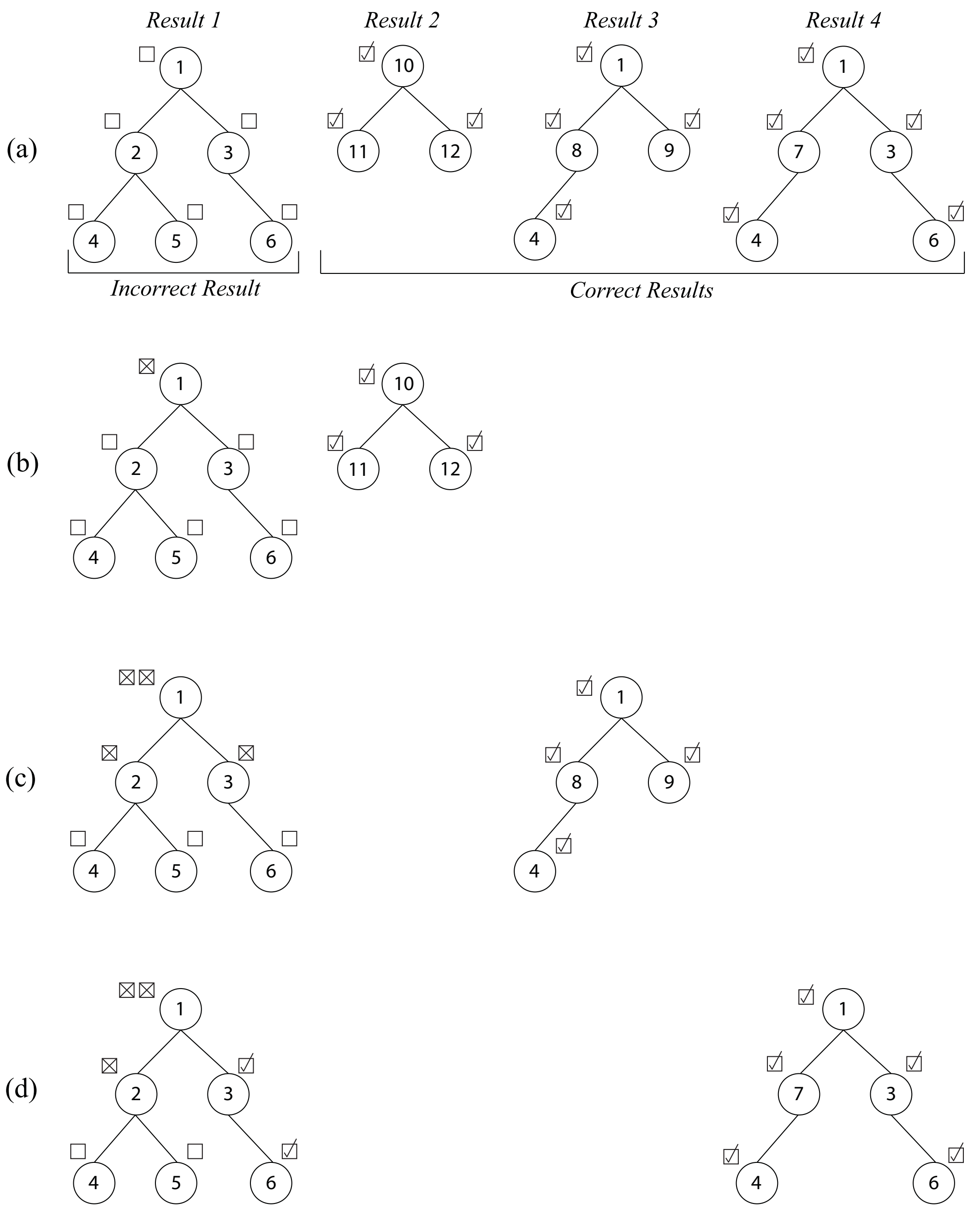


Figure 5.9: An example to automatic conversion of *Shallow Evaluation* input into *Deep Evaluation* input. **(a)** The initial situation of the parse trees associated with 4 translation results. *Result 1* is the only incorrect result, while *Result 2–4* are evaluated as correct by the user. At this point, all nodes of the incorrect result are initialized to □, while all nodes of the correct results are initialized to ☑. **(b)–(d)** The situation after each successive comparison of the incorrect result with one of the correct translation results. Note that changing the comparison order does not effect the final configurations of the parse trees.

5.3 Partially Matching Contexts

We have previously given CONFIDENCE-VALUE-EXACT in Algorithm 4, and described the way it is used to calculate the confidence value of a given translation result. In this section we will revise this procedure and apply some modifications.

In order for CONFIDENCE-VALUE-EXACT to use a co-occurrence rule in confidence value calculation of a translation result, a rule that matches the current subtree (the subtree in the parse tree of the translation result, that is currently processed) has to be available. Additionally, that rule should contain *a context that is identical to the context of the current subtree.* If such a rule exists, then the aggregate confidence factor associated with the matching context of the rule is returned immediately; otherwise the confidence value calculation continues recursively.

Requiring a rule-context, to match the context of the current subtree exactly is a constraint that is too strict. In this section, we relax this constraint in such a way, that in the absence of an exactly matching context, one or more *partially matching* contexts are used for deriving an aggregate confidence factor.

When we allow partial matching of contexts, we should first define a metric that reflects how close a given match is to a perfect one. Therefore, we define our metric, *match-ratio* as

$$match\_ratio(RC,\ TC) = \begin{cases} 1 & \text{if } length(RC) = 0, \\ \dfrac{matched(RC,\ TC)}{length(RC)} & \text{otherwise.} \end{cases} \tag{5.26}$$

where RC is the rule-context, TC is the context of the current subtree in the parse tree of the translation result, $length(RC)$ is the total number of the elements in RC and $matched(RC,\ TC)$ is the number of matched elements between RC and TC. Note that the match-ratio calculated for an empty rule-context is always 1.

Context matching is done simply by comparing the corresponding elements of given two contexts from left to right, i.e., from child to parent. For example, comparing the contexts $[4_{(2)},\ 6_{(1)},\ 7_{(4)},\ 8_{(2)}]$ and $[4_{(2)},\ 6_{(1)},\ 9_{(2)}]$ will yield two matching elements, $4_{(2)}$ and $6_{(1)}$.

The examples in this section use the context-dependent co-occurrence rule given below:

$$\begin{aligned} 1(2,\ 3) &- [4_{(2)},\ 5_{(3)}](0.3) \\ &- [4_{(2)},\ 6_{(1)},\ 7_{(4)},\ 8_{(2)}](0.7) \\ &- [4_{(2)},\ 9_{(1)},\ 10_{(2)}](0.9) \\ &- [4_{(2)},\ 12_{(1)}](0.4) \end{aligned} \tag{5.27}$$

The rule above is depicted graphically in Figure 5.10. This rule contains four contexts, namely $[4_{(2)},\ 5_{(3)}]$, $[4_{(2)},\ 6_{(1)},\ 7_{(4)},\ 8_{(2)}]$, $[4_{(2)},\ 9_{(1)},\ 10_{(2)}]$ and $[4_{(2)},\ 12_{(1)}]$ which are associated with four different aggregate confidence factors, 0.3, 0.7, 0.9 and 0.4, respectively.

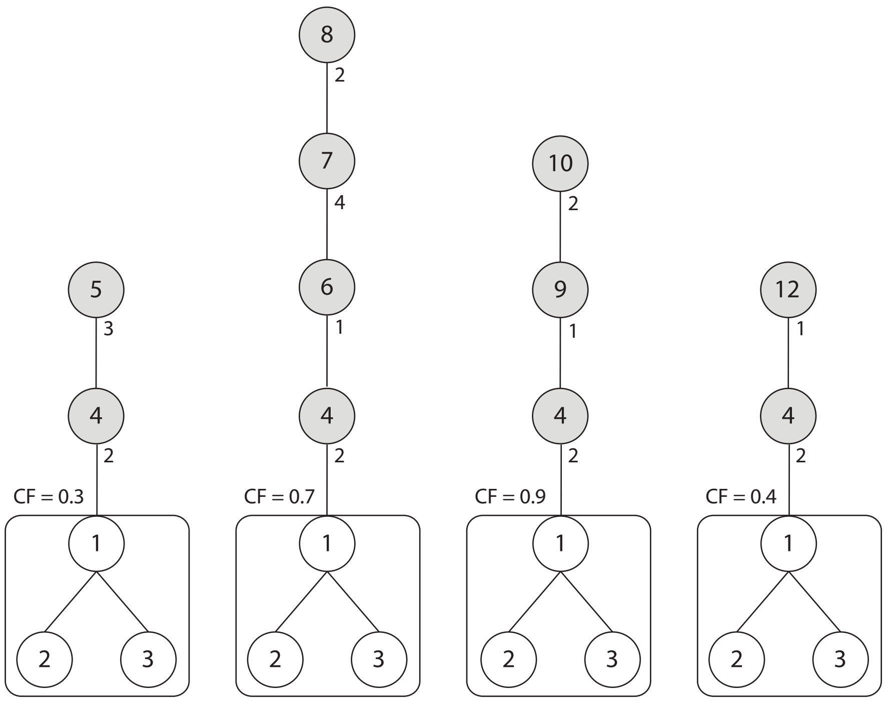


Figure 5.10: The Context-Dependent Co-occurrence Rule (5.27).

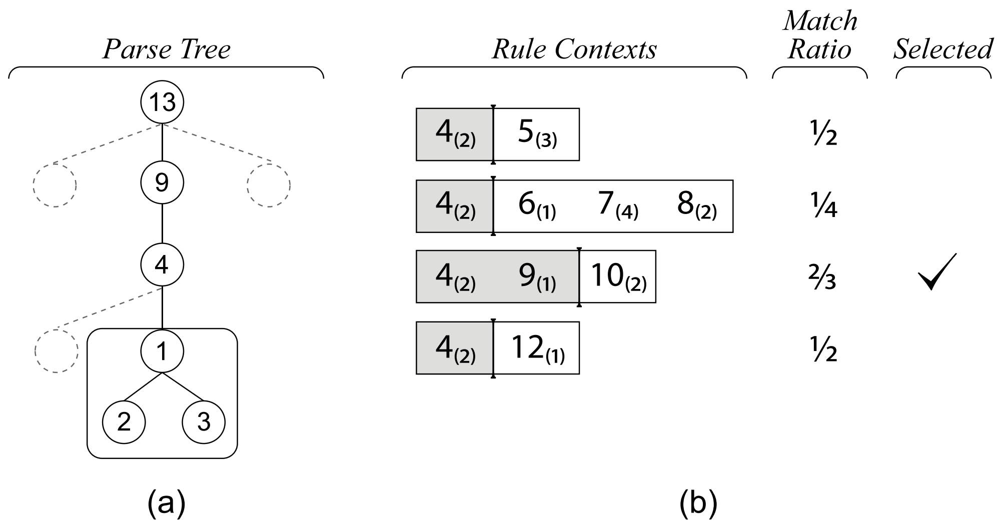


Figure 5.11: Partial Matching of Contexts: Case 1. **(a)** An example parse tree. A confidence value will be calculated for the subtree surrounded with the square. Nodes that are not important are drawn in dashed line pattern. **(b)** Third context has the highest match-ratio, therefore it is selected for confidence value calculation.

Given the current subtree and a rule that matches this subtree, we calculate an aggregate confidence value in three steps. In the first step we calculate match-ratios for all contexts available in the rule. Then we select a subset of the rule-contexts, elements of which are the best matching ones. Finally, we calculate an aggregate confidence factor using the selected subset.

A subset of rule-contexts, elements of which match the context of the given subtree best, is selected as follows:

- Case 1: If there is a unique rule-context with the highest non-zero match-ratio, then only that rule-context is selected. (For an example, see Figure 5.11)
- Case 2: If there are multiple rule-contexts with the highest non-zero match-ratio, then the longest one of those rule-contexts is selected. (For an example, see Figure 5.12)
- Case 3: If the longest rule-context is not unique, then all such rule-contexts are selected. (For an example, see Figure 5.13)

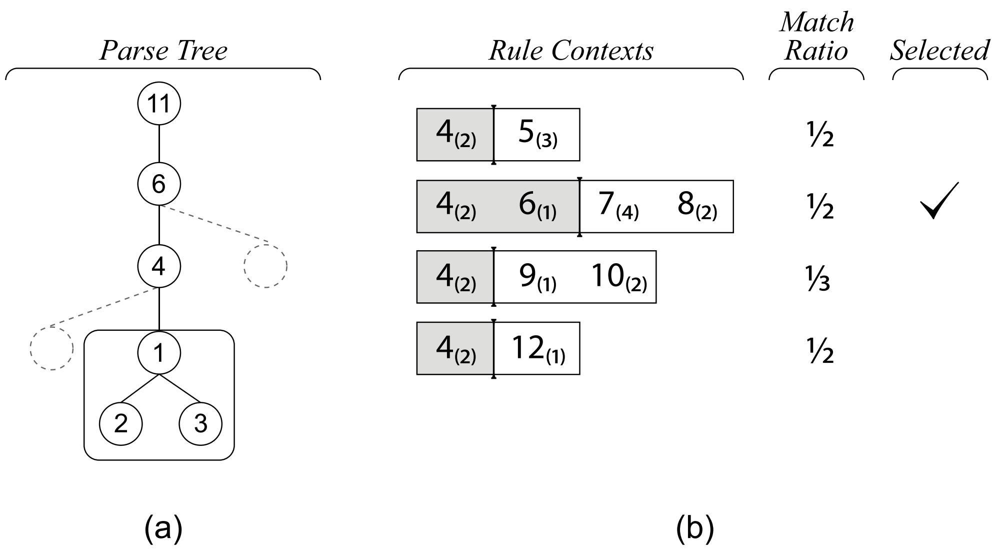


Figure 5.12: Partial Matching of Contexts: Case 2. **(a)** An example parse tree. A confidence value will be calculated for the subtree surrounded with the square. Nodes that are not important are drawn in dashed line pattern. **(b)** As there are multiple rule-contexts with the highest match-ratio, the longest one of those, which is the second rule-context, is selected.

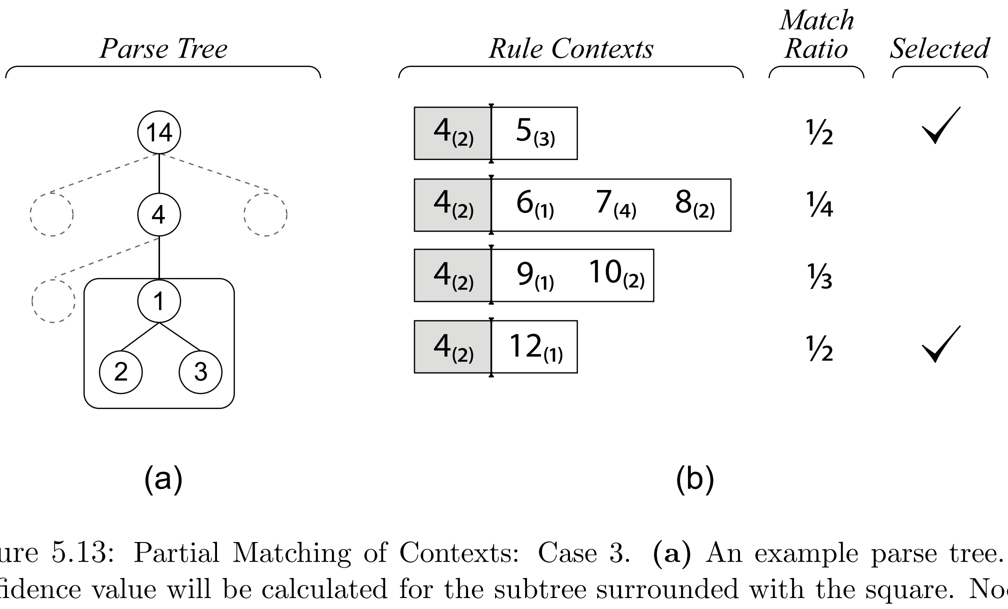


Figure 5.13: Partial Matching of Contexts: Case 3. **(a)** An example parse tree. A confidence value will be calculated for the subtree surrounded with the square. Nodes that are not important are drawn in dashed line pattern. **(b)** As there are two rule-contexts with the highest match-ratio, and the lengths of them are equal, both of the of the contexts are selected.

In the last step, the aggregate confidence factor for the current subtree — T — given the matching rule — R — is calculated as

$$ACF(T,R) = CV[T] + match\_ratio[S] \times \left(\frac{\sum\limits_{RC \in S} (ACF[RC])}{|S|} - CV[T] \right), \quad (5.28)$$

where $CV[T]$ is the original confidence value of T (calculated as the multiplication of the individual confidence factors of the templates in T), S is the selected subset of rule contexts, $match\_ratio[S]$ is the match-ratio of the rule-contexts in S (which is shared by all), and $ACF[RC]$ is the aggregate confidence factor associated with the rule-context RC. This formula is intuitive. The calculated aggregate confidence factor approaches to the original confidence value of the subtree, when match-ratio decreases. And as the match-ratio increases, it approaches to the average of the aggregate confidence factors associated with the rule-contexts in S.

For example, given that the original confidence value of the subtree 1(2, 3) in Figure 5.13(a) is 0.6, the aggregate confidence factor calculated for this subtree is

$$0.6 + 0.5 \times \left(\frac{0.3 + 0.4}{2} - 0.6 \right) = 0.475. \quad (5.29)$$

Up to now, we have studied the cases for which at least one rule-context has a non-zero match ratio. Another case is the one where a context-dependent co-occurrence rule matching the current subtree exists, but all of the contexts have a match ratio of zero. The naive solution is simply calculating the confidence value recursively without using the matching rule, if a rule-context with a non-zero match ratio is not available.

This approach may not satisfy user expectations. Consider a situation where the user dislikes a combination of templates. He evaluates that combination as incorrect, but the combination appears over and over in completely different contexts. We cannot expect a user to evaluate that combination for all possible contexts. Therefore — even if a non-zero rule-context does not exist — the previous evaluations should influence the confidence value calculated for the current subtree.

We achieve this effect by taking the average of the aggregate confidence factors of all rule-contexts, and the confidence value of the subtree is calculated recursively, as given in the equation below:

$$ACF(T,R) = \left(CV\_recursive[T] + \sum_{RC \in A} ACF[RC] \right) / \left(|A| + 1 \right). \tag{5.30}$$

In this equation $CV\_recursive[T]$ is the confidence value calculated for the current subtree T recursively, and A is the set that contains all rule-contexts in the rule.

The whole partial context matching process is formalized in Algorithm 9, which provides the procedure CONFIDENCE-VALUE-PARTIAL.

```
CONFIDENCE-VALUE-PARTIAL($node$)
 1: $tree \leftarrow$ the tree rooted at $node$
 2: $context \leftarrow$ the context of $node$
 3: $rule\_found \leftarrow$ false
 4: if there exists a co-occurrence rule $R$ that matches $tree$ then
 5:     Calculate match-ratio for all contexts in $R$.
 6:     Select the subset $S$, from the contexts in $R$, that best match $context$.
 7:     if $S \neq \emptyset$ then
 8:         //See Formula (5.28)
 9:         return $CV[T] + match\_ratio[S] \times \left( \frac{\sum_{RC \in S} (ACF[RC])}{|S|} - CV[T] \right)$
10:     else
11:         $rule\_found \leftarrow$ true
12:     end if
13: end if
14: //Calculate the confidence value recursively.
15: $confidence \leftarrow$ confidence factor of the template represented by $node$
16: $children \leftarrow \{child : child$ is a child of $node\}$
17: for all $child \in children$ do
18:     $confidence \leftarrow confidence \times$ CONFIDENCE-VALUE-PARTIAL($child$)
19: end for
20: if $rule\_found =$ true then
21:     //See Formula (5.30)
22:     $A \leftarrow \{RC : RC$ is a rule context in $R\}$
23:     return $\left( confidence + \sum_{RC \in A} ACF[RC] \right) / (|A| + 1)$
24: else
25:     return $confidence$
26: end if
```

Algorithm 9: CONFIDENCE-VALUE-PARTIAL. Returns the confidence value of a translation result.

Chapter 6

Test Results and Evaluation

In this chapter, we provide our results of the translation tests on the suggested morphological disambiguation and user evaluation mechanisms. For performance evaluation, we used two different metrics, which are *BLEU* and *P@n*.

Precision is a widely used metric for evaluating the system performance, especially in the field of Information Retrieval (IR). When precision is calculated at a given cut-off rank, only the topmost results returned by the system are considered. This measure is called "precision at n" or P@n, which is calculated as:

$$P@n = \frac{\#\text{ of correct results in top-}n\text{ translation results}}{n} \tag{6.1}$$

If for a certain translation, the number of translation results, k, is less than n, then n is taken as k. In our tests, we use the P@n measure for $n = 1, 3, 5$.

The second method, *Bilingual Evaluation Understudy*, (BLEU), measures the closeness of a translation result generated by a machine translation system to a correct translation reference by using n-gram based method. The next section reviews the BLEU method.

6.1 BLEU Method

BLEU Method is developed to obtain a standart method for evaluation of machine translation systems. Manual evaluation of MT systems is expensive in terms of time and effort needed to complete the task. BLEU is a quick, inexpensive, language-independent and automatic evaluation method, which correlates highly with human evaluation [30]. To judge the quality of a machine translation result, BLEU calculates its closeness to one or more reference human translations using n-grams. A BLEU score varies between 0 and 1, where a score of 1 denotes that the result is an exact translation.

In order to calculate a BLEU score for a candidate translation result, first the *modified n-gram precisions* for $n = 1 \ldots N$ are calculated. Then the geometric mean of the calcuated n-gram precisions is found. The calculated precision value can already distinguish between good and bad candidate results if the length of the candidate is equal or longer than that of the reference translations. In order to penalize candidates that are too short, BLEU also uses a multiplicative brevity penalty. A detailed description of the BLEU method can be found in [30].

In our experiments we take the candidate as the translation result with the highest confidence value. In cases where multiple results with the highest confidence value exist, candidate is taken as the first result generated. We use a single reference translation for each element in the testing subset. Also in our experiments, we used the parameter value $N = 4$, as recommended in [30].

6.2 Performance Tests

A data collection of 435 translation examples has been created for experimental evaluation (See Appendix D). This collection is divided into 3 subsets, sizes of which are shown in Table 6.1.

- Training Subset 1 is used for extracting the translation templates. When

Table 6.1: Sizes of the Translation Example Subsets.

	Size
Training Subset 1	315
Training Subset 2	20
Testing Subset	100
Total	**435**

we fed this set of translation examples to the learning algorithm, and let the algorithm run for 2 iterations, a total of 1776 translation templates were extracted; among which, 679 atomic and 1097 non-atomic templates were found.

- Training Subset 2 is used to train the system during Deep and Shallow Evaluations.
- Testing Subset contains the translation examples which were used later, during the performance evaluation of the system.

All three subsets contain unique elements, i.e., there exists no translation example that is shared by any two subsets; however, it is allowed for translation examples in the training and testing subsets to contain common substrings. In this way, we guarantee that the system is not directly trained for the elements of the testing subset, and thus avoid a flawed experiment. In the following subsections, we present the results of the tests conducted.

6.2.1 Tests on Morphological Disambiguation

We measured the effects of morphological disambiguation on the performance of the translation system. In this experiment we focused on Turkish to English translation, as we have a disambiguator only for Turkish.

Morphological disambiguation eliminates wrong morphological parses assigned to each word. Without disambiguation, the number of lexical-level representation possibilities for each translation input will be high. Disambiguation cuts the number of lexical-level representation possibilities, therefore reduces the time required to complete the translation. The results are given in Table 6.2.

Table 6.2: Effects of Morphological Disambiguation on Translation.

Disambiguation	**Avg. BLEU Score**	**Avg. P@1**	**Avg. P@3**	**Avg. P@5**	**Avg. # of lexical-form possibilities**	**Avg. Time (sec.)**
Disabled	85.4%	70.8%	74.1%	74.2%	10.2	82
Enabled	84.8%	68.7%	73.6%	74%	3.6	65

After morphological disambiguation, the average number of lexical-form possibilities per input decreased from 10.2 to 3.6. As a result, the average time required to complete a translation decreased from 82 seconds to 65 seconds. This approximately corresponds to an improvement of 25% in time consumption.

In our tests, the system extracted 1776 translation templates, but in a real-word application, the number of templates has to be in the order of millions. In our tests, we observed that for a lot of incorrect lexical-form possibilities, the Earley parser terminated at the first few steps, as no rule matching the incorrect terminal symbols were found among the known templates. We expect that the importance of morphological disambiguation will get higher as the number of the templates increases. This is because of the fact that as the number of translation templates increases, the Earley parser will be able to find some translation templates matching the first few tokens of the incorrect lexical-form possibilities. Therefore, the time required for the termination of the Earley parser become higher for the incorrect lexical-form possibilities as the number of translation templates increases.

Table 6.2 also shows us that the accuracy of the disambiguator is high. The improvements in efficiency stated above, can be achieved by sacrificing only a little from the translation effectiveness. Some amount of drop in the effectiveness is tolerable, as the disambiguation process can incorrectly eliminate some of the correct lexical-form possibilities. For our system, the BLEU score drops only

0.4% points and P@1 drops by 2% when the disambiguator is turned on.

As discussed above, the number of templates used in our tests is small compared to the number required by a real-word application. As the number of templates increases, contrary to what is observed in this test, we expect that the performance attained by disambiguation will be higher than it is attained without disambiguation. This is because of the fact that incorrect lexical-form possibilities can result in the generation of wrong translations results if the templates are general enough and numerous.

6.2.2 Tests on Deep and Shallow Evaluation

In order to measure the performance of Deep and Shallow Evaluation methods, we trained the system using the translation examples in Training Subset 2. We ran the translation algorithm for each element of this set, first in English to Turkish direction, and then performed the Deep Evaluation. Namely, whenever we disliked the ordering of the translation results, we marked the erroneous nodes in the translation parse trees of the incorrect results and the root nodes of the correct results as well. Then, the same process was repeated in the reverse direction of translation. When the Deep Evaluation for all elements of the Training Set 2 was finished, the Shallow Evaluation was applied in a similar fashion, marking the correct and incorrect translation results.

The summary of the user evaluation processes are given in 6.3. As Deep Evaluation is a much more detailed process compared to Shallow Evaluation, the former takes approximately twice as much time as the latter. Furthermore, in Deep Evaluation, the user has greater control on the process; as a result, the number of context-dependent co-occurrence rules learned in Deep Evaluation is less than the rules learned in Shallow Evaluation.

Table 6.4 presents the results of the tests done in the English to Turkish direction of translation. In this direction, initially the average BLEU score was 90.6%. When the context-dependent co-occurrence rules learned from Deep Evaluation

Table 6.3: Summary of the Deep and Shallow Evaluation.

	# of Co-occurrence Rules Learned		Duration of the Evaluation (min.)
	EN to TR	**TR to EN**	
Shallow Evaluation	60	69	7:05
Deep Evaluation	44	50	13:30

were used in the ranking of the translation results, the BLEU score increased to 94.6%. Likewise the average P@1 (precision at the top-1 results) increased from 75% to 87%, while the average P@3 increased from 82.3% to 83.7%. Also, there was a marginal increase in the average P@5 value. For this experiment, using the rules learned from Shallow Evaluation resulted in exactly the same performance improvements. Table 6.5 shows the distribution of the position of the first correct result among the generated results.

Table 6.4: Experimental Results for English to Turkish Translation.

	Average BLEU Score	**Average P@1**	**Average P@3**	**Average P@5**
Initial	90.6%	75%	82.3%	81.7%
Shallow Evaluation	94.6%	87%	83.7%	82.1%
Deep Evaluation	94.6%	87%	83.7%	82.1%

Table 6.5: Position of the First Correct Result for English to Turkish Translation.

	1	**2-3**	**4-5**
Initial	75%	21%	2%
Shallow Evaluation	87%	9%	2%
Deep Evaluation	87%	9%	2%

In Turkish to English direction as given in Table 6.6, initially the average BLEU score was 85.3%. When the context-dependent co-occurrence rules learned from Shallow Evaluation were used in the ranking of the translation results, the BLEU score increased to 87.9%. Similarly the average P@1 increased from 70%

to 79%, P@3 increased from 74.2% to 74.3%, and P@5 increased from 74.4% to 74.7%.

Results for Deep Evaluation were better. When the context-dependent co-occurrence rules learned from Deep Evaluation were used in the ranking of the translation results, the BLEU score increased to 88.9%. Similarly the average P@1, P@3 and P@5 increased to 81%, 75% and 74.7%, respectively. Table 6.7 shows the distribution of the position of the first correct result among the generated results.

Table 6.6: Experimental Results for Turkish to English Translation.

	Average BLEU Score	**Average P@1**	**Average P@3**	**Average P@5**
Initial	85.3%	70%	74.2%	74.4%
Shallow Evaluation	87.9%	79%	74.3%	74.7%
Deep Evaluation	88.9%	81%	75%	74.7%

Table 6.7: Position of the First Correct Result for Turkish to English Translation.

	1	**2-3**	**4-5**
Initial	70%	21%	4%
Shallow Evaluation	79%	13%	3%
Deep Evaluation	81%	11%	3%

In Turkish to English direction, results for the Deep Evaluation were better than that of the Shallow Evaluation. This is because of the fact that, in Deep Evaluation the user can fine-tune the templates that will be learned from the evaluation, while this is not possible in Shallow Evaluation. Therefore in the general case, we expect the number of incorrect context-dependent co-occurrence rules learned by the Shallow Evaluation to be higher. Also, as the number of rules learned by Deep Evaluation is usually less than it is for Shallow Evaluation, the time consumption of the ranking process will also be lower if the former approach is followed. However, we expect that the users will prefer Shallow Evaluation, due to its simplicity.

Chapter 7

Conclusion

In this thesis, we added several new modules to an existing example-based machine translation system, extending its capabilities. The major contribution of this work is an improved ranking mechanism for the translation results that learns gradually from user feedback (see Chapter 5). After a translation, the user always has the option of evaluating the generated results. From the evaluaton, the system learns context-dependent co-occurrence rules which may be consulted in the results ordering phases of the upcoming translations.

In order to sort the translation results, the earlier versions of the system solely used the confidence factors associated with each template. Confidence factors were calculated in the learning phase once, and never updated thereafter. In our approach, confidence factor scheme is improved by the inclusion of context-dependent co-occurrence rules. With each user evaluation in the translation phase, the system continues to learn context-dependent co-occurrence rules.

Certain translation templates may be assigned low confidence factors when considered individually, but their co-existence in a translation result may deserve a higher confidence. The reverse can also be true. The original confidence factor assignment scheme did not handle template combinations, but considered each template individually. In our approach, the user has the chance to influence the confidence values of translation template combinations, without affecting the original confidence factors that will be used when the templates are utilized individually.

The system provides two different interfaces for inputting user feedback. In the Shallow Evaluation interface, the user simply marks correct and incorrect translations. On the other hand, in the Deep Evaluation, as the name implies, the user can evaluate individual nodes of the parse trees associated with each translation result, where each node represents a separate translation template. Therefore, Deep Evaluation takes more time, as it requires more attention and expertise. However, Deep Evaluation provides fine-tuning capabilities which are not offered by the Shallow Evaluation.

In our tests, we observed significant performance improvements in the average BLEU scores and precision values at the top results. In Turkish to English direction, the improvements for the Deep Evaluation were better than those of the Shallow Evaluation, as expected. The achieved performance improvements, which need to be confirmed by further tests on a larger corpus, were promising.

In Section 5.1, the context of a subtree in the parse tree of a translation result (where the subtree corresponds to a phrase in the translation) was defined as a chain of nodes. This abstract definition allowed us to develop a context matching algorithm (see Section 5.3) in a simple manner. However, we expect a linguistically influenced definition to be superior, as a more natural definition of the context that a phrase occurs in would be based on the words surrounding that phrase.

Another extension proposed for the translation system described, is a rule-based morphological disambiguator for Turkish (see Chapter 4). A morphological disambiguator, identifies and removes incorrect morphological parses of the words in a given sentence. If the number of translation templates kept in the system is high, ambiguity in the input may increase the number of incorrect translation results and slow down the translation process. The sentences in agglutinative languages such as Turkish tend to be more ambiguous. Therefore, for a machine translation system, where one of the languages is Turkish, morphological disambiguation becomes more critical. In our tests, we observed that the morphological disambiguator reduces the average time required for the translation of a given input.

To conclude, we must repeat that language is a complex human phenomenon — even the language acquisition mechanisims of children are still not completely understood. Hence, the future developments in the field of machine translation will depend, among several other factors, on our ability to understand and simulate the higher functions of our brain. New discoveries and the progress in computer science and technology, especially those in the subfield of artificial intelligence, would certainly improve the capabilities of future machine translation systems. However, for the time being, the difficulties of machine translation remind us of the Italian saying: *"Traduttore, traditore!"* (Translator, you're a traitor!) [18].

Bibliography

[1] K. Altıntaş, F. Can, and J. M. Patton. Language change quantification using time-separated parallel translations. Literary & Linguistic Computing (accepted, but not yet published).

[2] E. L. Antworth. *PC-Kimmo: A two level processor for morphological analysis.* Summer Institute of Linguistics, Dallas, Texas, 1990.

[3] E. Brill. A simple rule-based part-of-speech tagger. In *Proceedings of the third Conference on Applied Natural Language Processing*, pages 152–155, Trento, Italy, March 31 - April 3, 1992.

[4] J. F. Bullock. *Preaching with a cupped ear: Hans-Georg Gadamer's philosophical hermeneutics as postmodern wor(l)d.* New York: P. Lang, 1999.

[5] İ. Çiçekli. Inducing translation templates with type constraints. In *Proceedings of Example-Based Machine Translation Workshop, MT Summit X*, pages 27–34, Phuket, Thailand, September 2005.

[6] İ. Çiçekli and H. A. Güvenir. Learning translation templates from bilingual translation examples. In M. Carl and A. Way, editors, *Recent Advances in Example-Based Machine Translation*, pages 247–278. Kluwer Academic Publishers, Boston, USA, 2003.

[7] N. Chomsky. *Aspects of the theory of syntax.* MIT Press, 1965.

[8] K. W. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the second Conference on Applied Natural*

Language Processing, pages 136–143, Austin, Texas, USA, February 09-12, 1988.

[9] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part-of-speech tagger. Technical report, Xerox Palo Alto Research Center, Palo Alto, CA, USA, 1992.

[10] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part-of-speech tagger. In *Proceedings of the third Conference on Applied Natural Language Processing*, pages 133–140, Trento, Italy, March 31 - April 3, 1992.

[11] R. de Pedro. The translatability of texts: A historical overview. *Meta: Journal des traducteurs*, 44(4):546–559, 1999.

[12] W. Dilthey. *Gesammelte Schriften.* Göttingen: Vanderhoeck & Ruprecht, 1961-2001.

[13] H. Doğan. Example based machine translation with type associated translation templates. Master's thesis, Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey, 2007.

[14] M. Fram-Cohen. Reality, language, translation: What makes translation possible. In *Proceedings of American Translators Association 26th Annual Conference*, Miami, Florida, 1985.

[15] H. C. Hanko. Issues in hermeneutics. *Protestant Reformed Theological Journal*, 1991.

[16] M. Heidegger. *Being and Time.* Blackwell Publishers, 1997.

[17] Ö. İstek. A link grammar for turkish. Master's thesis, Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey, 2006.

[18] R. Jakobson. On linguistic aspects of translation. In R. A. Brower, editor, *On Translation*, pages 232–239. Harvard University Press, Cambridge, MA, 1959.

[19] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

[20] İ. Kuruöz. Tagging and morphological disambiguation of turkish texts. Master's thesis, Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey, 1994.

[21] R. A. Makkreel. *Dilthey: Philosopher of the Human Studies.* Princeton University Press, 1993.

[22] J. C. Mallery, R. Hurwitz, and G. Duffy. Hermeneutics: from textual explication to computer understanding. In S. Shapiro, editor, *The Encyclopedia of Artificial Intelligence.* John Wiley and sons, New York, 1987.

[23] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, Massachusetts, 1999.

[24] M. Nagao. A framework of a mechanical translation between japanese and english by analogy principle. In *Proc. of the international NATO symposium on Artificial and human intelligence*, pages 173–180, New York, NY, USA, 1984. Elsevier North-Holland, Inc.

[25] K. Oflazer. Two-level description of turkish morphology. *Literary and Linguistic Computing*, 9(2):137–148, 1994.

[26] K. Oflazer and İ. Kuruöz. Tagging and morphological disambiguation of turkish text. In *Proceedings of the fourth Conference on Applied Natural Language Processing*, pages 144–149, Stuttgart, Germany, October 1994.

[27] K. Oflazer and G. Tür. Combining hand-crafted rules and unsupervised learning in constraint-based morphological disambiguation. In *Proceedings of the ACL-SIGDAT Conference on Empirical Methods in Natural Language Processing*, Philadelphia, PA, USA, May 1996.

[28] K. Oflazer and G. Tür. Morphological disambiguation by voting constraints. In P. R. Cohen and W. Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth*

Conference of the European Chapter of the Association for Computational Linguistics, pages 222–229, Somerset, New Jersey, 1997. Association for Computational Linguistics.

[29] Z. Öz and İ. Çiçekli. Ordering translation templates by assigning confidence factors. In *AMTA '98: Proceedings of the Third Conference of the Association for Machine Translation in the Americas on Machine Translation and the Information Soup*, pages 51–61, London, UK, 1998. Springer-Verlag.

[30] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, July 2002.

[31] G. Steiner. *After Babel. Aspects of Language and Translation.* Oxford University Press, Oxford, 1975.

[32] G. Tür. Using multiple sources of information for constraint-based morphological disambiguation. Master's thesis, Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey, 1996.

[33] B. Vauquois. A survey of formal grammars and algorithms for recognition and transformation in mechanical translation. In *IFIP Congress (2)*, pages 1114–1122, 1968.

[34] W. Weaver. Translation. In W. N. Locke and A. D. Boothe, editors, *Machine Translation of Languages*, pages 15–23. MIT Press, Cambridge, MA, 1949/1955. Reprinted from a memorandum written by Weaver in 1949.

Appendix A

A Deep Evaluation Example

This appendix provides a visual walkthrough for the Deep Evaluation of the translation results produced for the input phrase

$$\text{“sarı saçlı kadın”,} \tag{A.1}$$

which was used as an example previously in Chapter 5. The system had returned two translation results, which were:

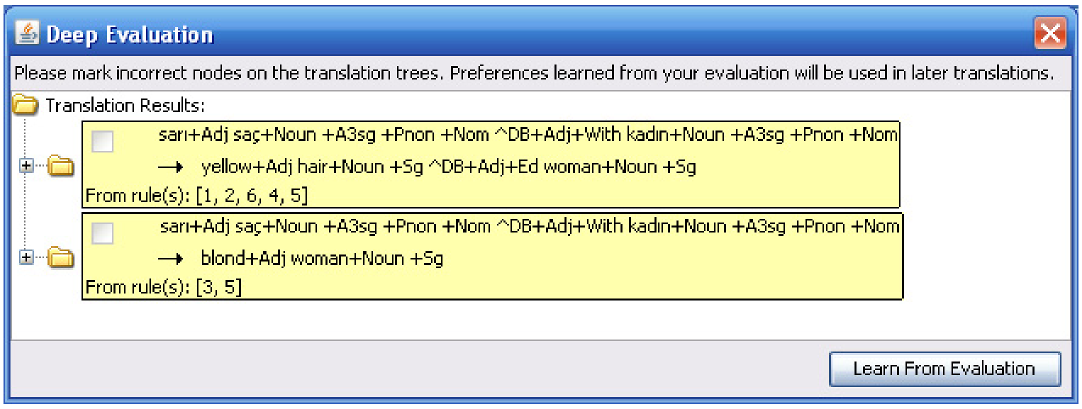
$$\begin{aligned}&\text{“yellow haired woman”} \\ &\text{“blond woman”.}\end{aligned} \tag{A.2}$$

The parse trees of these translation results were given in Figure 5.4.

During the evaluation, each node in the parse tree is labeled with the partial translation implied by that node. The partial translations are given in the lexical-level. The parse tree structure of a translation result is given as a list in which the hierarchy of the nodes is denoted using indentation, where the indentation of a child node is more than that of its parent.

Figures A.1 through A.6 depict the Deep Evaluation steps followed by the user. The process is described step-by-step below:

- *Figure A.1:* Initially, only the root nodes are shown to the user, i.e., the trees are collapsed. Both of the nodes are marked as □.
- *Figure A.2:* The root node of the first translation result is set to ⊠. The color of the node is changed to red in order to denote that the translation implied by this node is incorrect. As the node is incorrect, it is expanded, revealing its children.
- *Figure A.3:* In the parse tree of the first translation result, Node 2 is set to state ⊠, therefore it is expanded and its children, Node 6 and Node 4, become visible. This also affects the state of its parent, Node 1, which is automatically set to the ⊠⊠ state. The user also sets the state of Node 5, which is the second children of the root node, to ☑.
- *Figure A.4:* As the error in the first translation result is isolated in the subtree rooted at Node 2, the user sets the state of Node 1 to ⊠☑. Now the background color of the node is green, but the erroneous portions of the translation, which are due to the erroneous subtree rooted at Node 2, are highlighted in red color.
- *Figure A.5:* The user evaluates Node 6 and Node 4. Node 6 implies the partial translation "sarı+Adj $\rightarrow$ yellow+Adj". Using this partial translation in the context of $[2_{(1)}, 1_{(1)}]$ is not wrong. Similarly, Node 4 could well be used in the same context correctly if Node 6 was not there. In other words, the cause of the error is using Nodes 6 and 4 together. When considered separately, using these nodes in the context they appear is not wrong. So, the states of both of the nodes are set to ☑ by the user.
- *Figure A.6:* The user marks the root node of the second translation result as ☑, since it is a correct translation. Therefore, the color of the node is set to green. The evaluation is completed now.



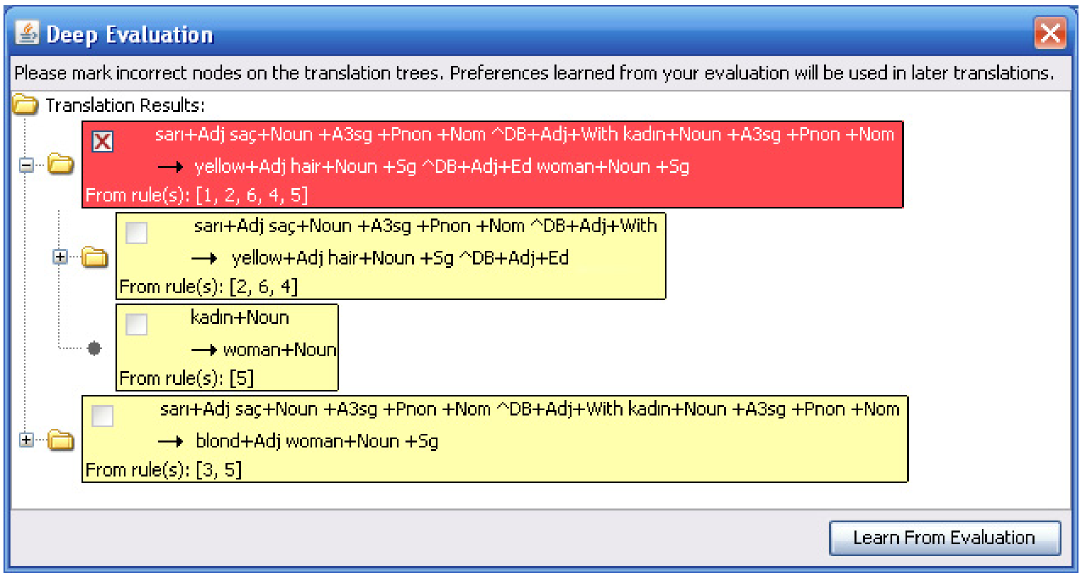
Please mark incorrect nodes on the translation trees. Preferences learned from your evaluation will be used in later translations.

Translation Results:

sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With kadın+Noun +A3sg +Pnon +Nom
→ yellow+Adj hair+Noun +Sg ^DB+Adj+Ed woman+Noun +Sg
From rule(s): [1, 2, 6, 4, 5]

sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With kadın+Noun +A3sg +Pnon +Nom
→ blond+Adj woman+Noun +Sg
From rule(s): [3, 5]

Figure A.1: 1st Step in the Deep Evaluation of the Results.



Please mark incorrect nodes on the translation trees. Preferences learned from your evaluation will be used in later translations.

Translation Results:

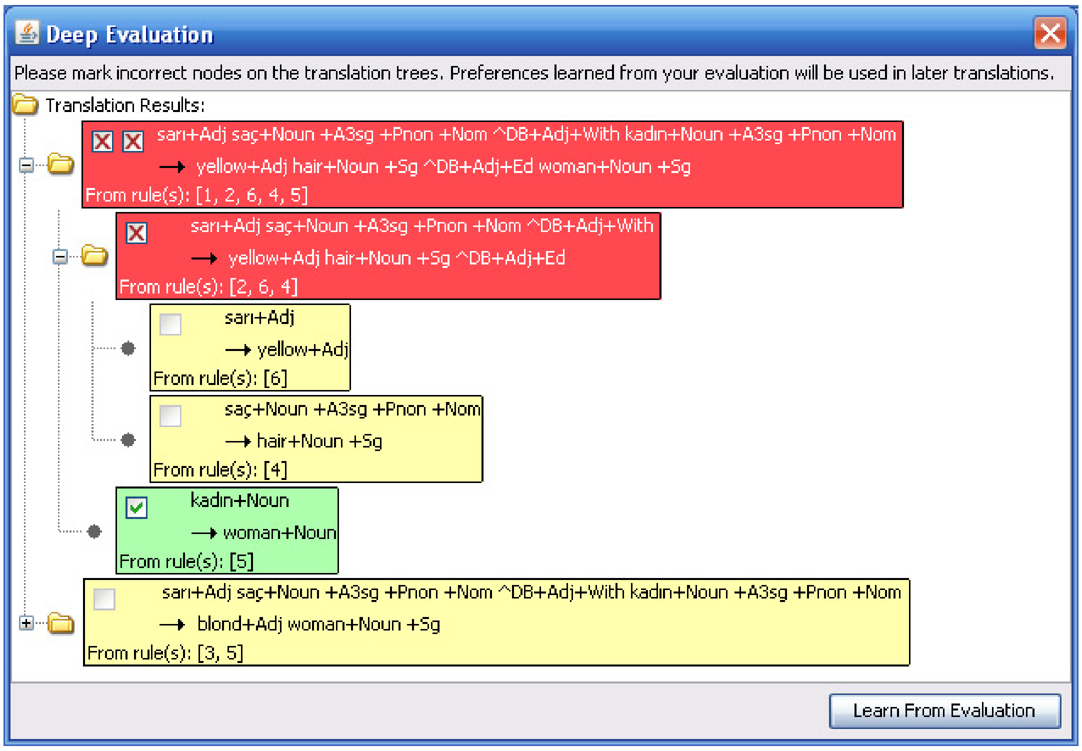
sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With kadın+Noun +A3sg +Pnon +Nom
→ yellow+Adj hair+Noun +Sg ^DB+Adj+Ed woman+Noun +Sg
From rule(s): [1, 2, 6, 4, 5]

sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With
→ yellow+Adj hair+Noun +Sg ^DB+Adj+Ed
From rule(s): [2, 6, 4]

kadın+Noun
→ woman+Noun
From rule(s): [5]

sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With kadın+Noun +A3sg +Pnon +Nom
→ blond+Adj woman+Noun +Sg
From rule(s): [3, 5]

Figure A.2: 2nd Step in the Deep Evaluation of the Results.



Deep Evaluation

Please mark incorrect nodes on the translation trees. Preferences learned from your evaluation will be used in later translations.

Translation Results:

☒ ☒ sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With kadın+Noun +A3sg +Pnon +Nom
→ yellow+Adj hair+Noun +Sg ^DB+Adj+Ed woman+Noun +Sg
From rule(s): [1, 2, 6, 4, 5]

☒ sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With
→ yellow+Adj hair+Noun +Sg ^DB+Adj+Ed
From rule(s): [2, 6, 4]

☐ sarı+Adj
→ yellow+Adj
From rule(s): [6]

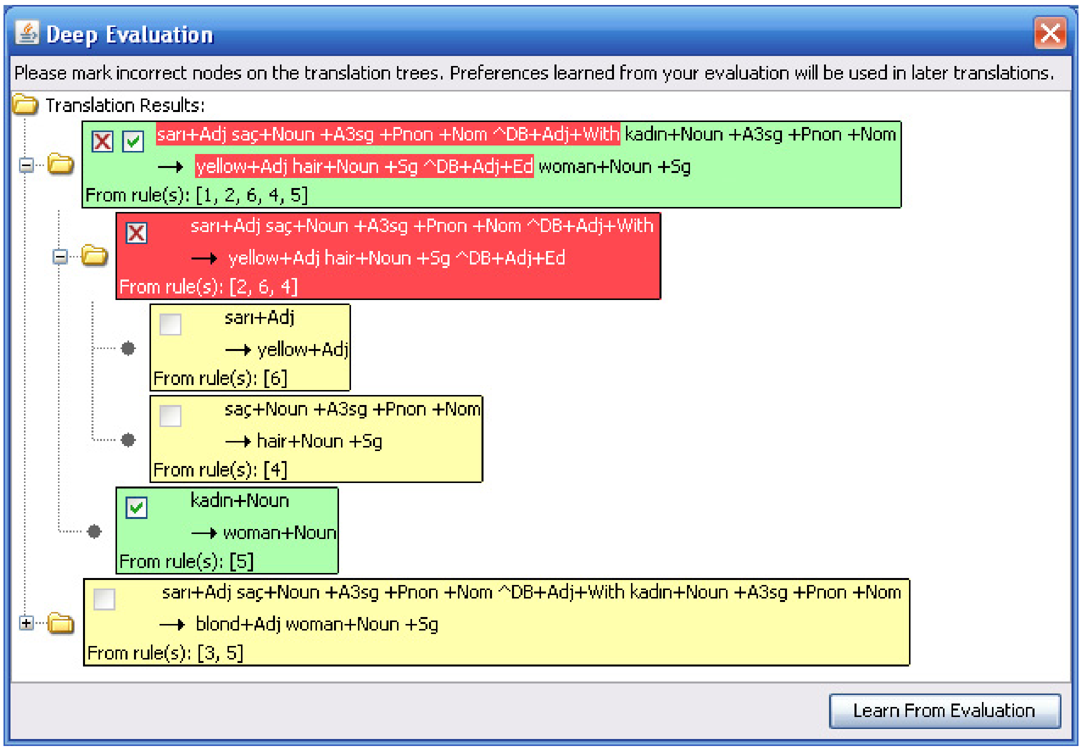
☐ saç+Noun +A3sg +Pnon +Nom
→ hair+Noun +Sg
From rule(s): [4]

☑ kadın+Noun
→ woman+Noun
From rule(s): [5]

☐ sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With kadın+Noun +A3sg +Pnon +Nom
→ blond+Adj woman+Noun +Sg
From rule(s): [3, 5]

Learn From Evaluation

Figure A.3: 3rd Step in the Deep Evaluation of the Results.



Deep Evaluation

Please mark incorrect nodes on the translation trees. Preferences learned from your evaluation will be used in later translations.

Translation Results:

☒ ☑ sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With kadın+Noun +A3sg +Pnon +Nom
→ yellow+Adj hair+Noun +Sg ^DB+Adj+Ed woman+Noun +Sg
From rule(s): [1, 2, 6, 4, 5]

☒ sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With
→ yellow+Adj hair+Noun +Sg ^DB+Adj+Ed
From rule(s): [2, 6, 4]

☐ sarı+Adj
→ yellow+Adj
From rule(s): [6]

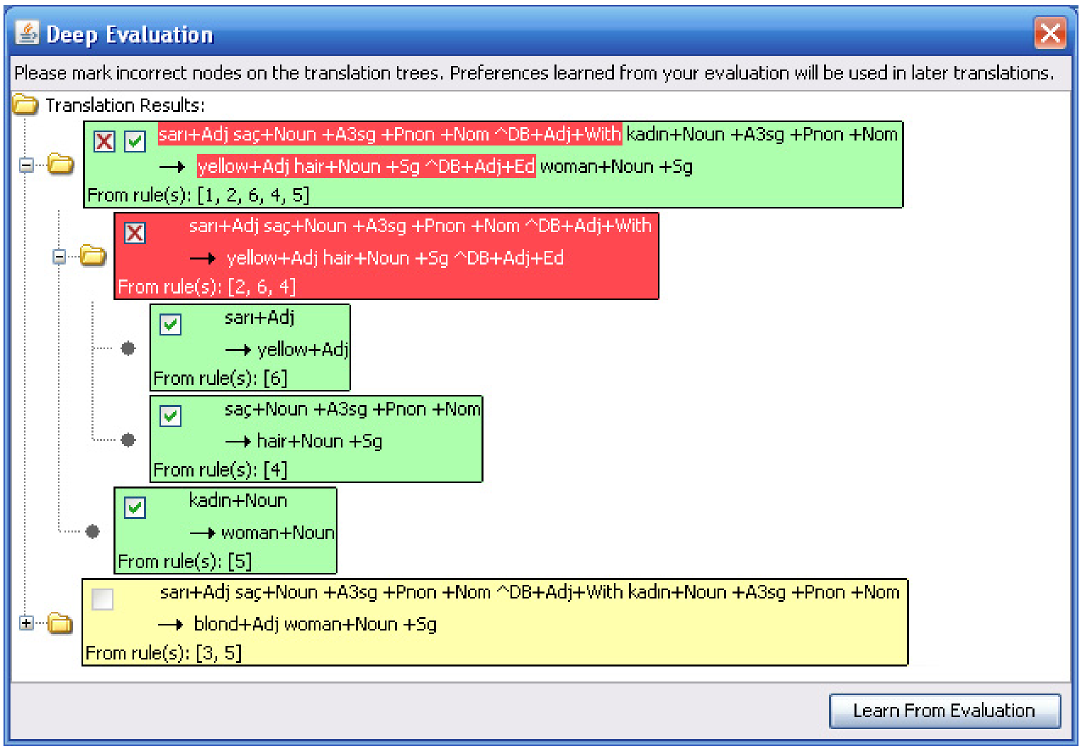
☐ saç+Noun +A3sg +Pnon +Nom
→ hair+Noun +Sg
From rule(s): [4]

☑ kadın+Noun
→ woman+Noun
From rule(s): [5]

☐ sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With kadın+Noun +A3sg +Pnon +Nom
→ blond+Adj woman+Noun +Sg
From rule(s): [3, 5]

Learn From Evaluation

Figure A.4: 4th Step in the Deep Evaluation of the Results.



Please mark incorrect nodes on the translation trees. Preferences learned from your evaluation will be used in later translations.

Translation Results:

☒ ☑ sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With kadın+Noun +A3sg +Pnon +Nom
→ yellow+Adj hair+Noun +Sg ^DB+Adj+Ed woman+Noun +Sg
From rule(s): [1, 2, 6, 4, 5]

☒ sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With
→ yellow+Adj hair+Noun +Sg ^DB+Adj+Ed
From rule(s): [2, 6, 4]

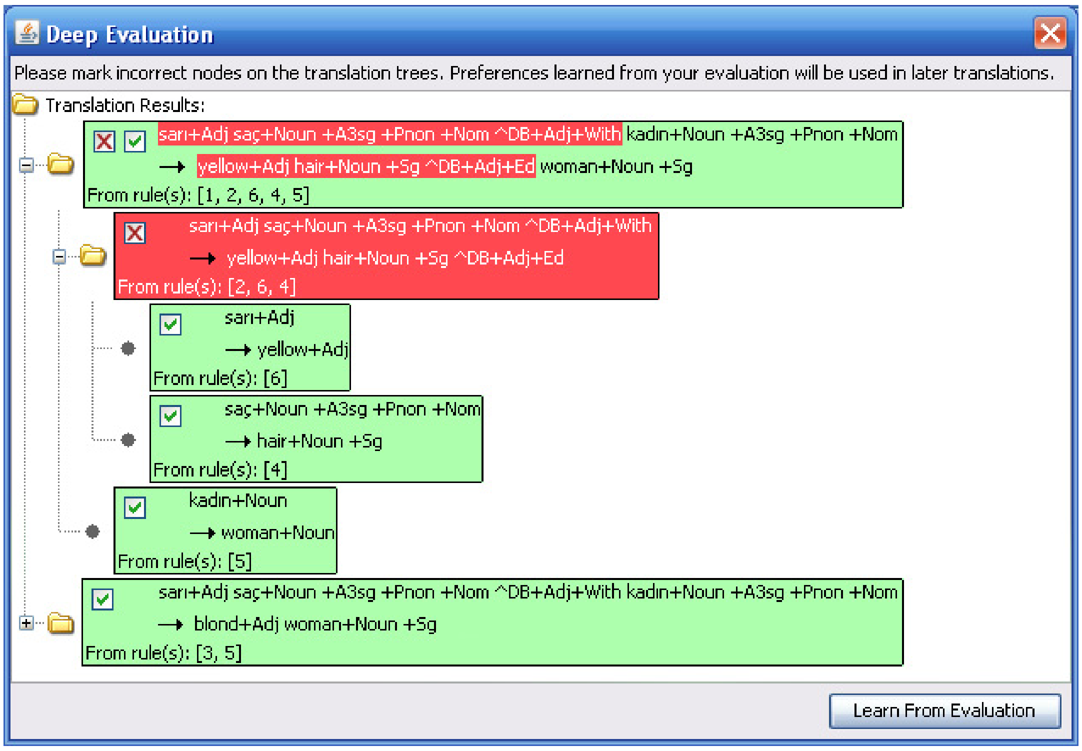
☑ sarı+Adj
→ yellow+Adj
From rule(s): [6]

☑ saç+Noun +A3sg +Pnon +Nom
→ hair+Noun +Sg
From rule(s): [4]

☑ kadın+Noun
→ woman+Noun
From rule(s): [5]

☐ sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With kadın+Noun +A3sg +Pnon +Nom
→ blond+Adj woman+Noun +Sg
From rule(s): [3, 5]

Figure A.5: 5th Step in the Deep Evaluation of the Results.



Please mark incorrect nodes on the translation trees. Preferences learned from your evaluation will be used in later translations.

Translation Results:

☒ ☑ sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With kadın+Noun +A3sg +Pnon +Nom
→ yellow+Adj hair+Noun +Sg ^DB+Adj+Ed woman+Noun +Sg
From rule(s): [1, 2, 6, 4, 5]

☒ sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With
→ yellow+Adj hair+Noun +Sg ^DB+Adj+Ed
From rule(s): [2, 6, 4]

☑ sarı+Adj
→ yellow+Adj
From rule(s): [6]

☑ saç+Noun +A3sg +Pnon +Nom
→ hair+Noun +Sg
From rule(s): [4]

☑ kadın+Noun
→ woman+Noun
From rule(s): [5]

☑ sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With kadın+Noun +A3sg +Pnon +Nom
→ blond+Adj woman+Noun +Sg
From rule(s): [3, 5]

Figure A.6: 6th Step in the Deep Evaluation of the Results.

Appendix B

English Suffixes

Inflectional and derivational suffixes recognized by the English morphological analyzer are given in Tables B.1 and B.2, respectively. One example is provided for each suffix. Note that the current version of the morphological analyzer does not handle prefixes. Therefore, prefixed forms of the words have to be added to the lexicon files, as if they were root words.

Table B.1: English Inflectional Suffixes.

Suffix	**Example**	
	Surface-Level	**Lexical-Level**
-'s	John's	john+Noun+Prop+Sg+Part+Gen
-ed	handled	handle+Verb+PastSimp+123SP
-er	stronger	strong+Adj+Comp
-est	brightest	bright+Adj+Sup
-ing	singing	sing+Verb+Prog
-s (Tense)	goes	go+Verb+Pres+3sg
-s (Plural)	houses	house+Noun+Pl

Table B.2: English Derivational Suffixes.

Suffix	**Example**	
	Surface-Level	**Lexical-Level**
-able	acceptable	accept+Verb+Inf^DB+Adj+Able
-age	leakage	leak+Verb+Inf^DB+Noun+Age+Sg
-al	experimental	experiment+Verb+Inf^DB+Noun+Al+Sg
-ance	aberrance	aberrant+Adj^DB+Noun+Ance+Sg
-ant	assistant	assist+Verb+Inf^DB+Noun+Ant+Sg
-ar	angular	angle+Noun+Sg^DB+Adj+Ar
-ary	adversary	adverse+Adj^DB+Noun+Ary+Sg
-ate	passionate	passion+Noun+Sg^DB+Adj+Ate
-ation	examination	examine+Verb+Inf^DB+Noun+Ation+Sg
-ative	talkative	talk+Verb+Inf^DB+Adj+Ative
-atory	perspiratory	perspire+Verb+Inf^DB+Adj+Atory
-cy	adequacy	adequate+Adj^DB+Noun+Cy+Sg
-ed	haired	hair+Noun+Sg^DB+Adj+Ed
-ee	employee	employ+Verb+Inf^DB+Noun+Ee+Sg
-eer	engineer	engine+Noun+Sg^DB+Noun+Eer+Sg
-en	golden	gold+Noun+Sg^DB+Adj+En
-ence	adolescence	adolescent+Adj^DB+Noun+Ence+Sg
-ent	dependent	depend+Verb+Inf^DB+Adj+Ent
-er	driver	drive+Verb+Inf^DB+Noun+Er+Sg
-ery	bakery	bake+Verb+Inf^DB+Noun+Ery+Sg
-ess	hostess	host+Noun+Sg^DB+Noun+Ess+Sg
-ful	joy	joy+Noun+Sg^DB+Adj+Ful
-ible	convertible	convert+Verb+Inf^DB+Adj+Ible
-ic	satanic	satan+Noun+Sg^DB+Adj+Ic
-ify	purify	pure+Adj^DB+Verb+Ify+Inf
-ion	interruption	interrupt+Verb+Inf^DB+Noun+Ion+Sg
-ise	apologise	apology+Noun+Sg^DB+Verb+Ise+Inf
-ish	childish	child+Noun+Sg^DB+Adj+Ish
-ism	dogmatism	dogma+Noun+Sg^DB+Noun+Ism+Sg
-ist	colonist	colony+Noun+Sg^DB+Noun+Ist+Sg
-ite	urbanite	urban+Adj^DB+Noun+Ite+Sg
-itive	acquisitive	acquire+Verb+Inf^DB+Adj+Itive
-ity	brevity	brief+Adj^DB+Noun+Ity+Sg
-ive	apprehensive	apprehend+Verb+Inf^DB+Adj+Ive
-ize	apologize	apology+Noun+Sg^DB+Verb+Ize+Inf

Table B.2: English Derivational Suffixes (Continued).

Suffix	**Example**	
	Surface-Level	**Lexical-Level**
-less	endless	end+Noun+Sg^DB+Adj+Less
-let	piglet	pig+Noun+Sg^DB+Noun+Let+Sg
-ling	duckling	duck+Noun+Sg^DB+Noun+Ling+Sg
-ly	friendly	friend+Noun+Sg^DB+Adj+Ly
-ment	development	develop+Verb+Inf^DB+Noun+Ment+Sg
-ness	kindness	kind+Adj^DB+Noun+Ness+Sg
-or	disambiguator	disambiguate+Verb+Inf^DB+Noun+Or+Sg
-ory	sensory	sense+Noun+Sg^DB+Adj+Ory
-ous	prestigious	prestige+Noun+Sg^DB+Adj+Ous
-sion	decision	decide+Verb+Inf^DB+Noun+Sion+Sg
-ster	songster	song+Noun+Sg^DB+Noun+Ster+Sg
-th	width	wide+Adj^DB+Noun+Th+Sg
-tion	abolition	abolish+Verb+Inf^DB+Noun+Tion+Sg
-ty	cruelty	cruel+Adj^DB+Noun+Ty+Sg
-ure	enclosure	enclose+Verb+Inf^DB+Noun+Ure+Sg
-y	rainy	rain+Noun+Sg^DB+Adj+Y

Appendix C

Lattice Structure for English

The lattice structure for English used in our translation system is given in Table C.1. The categories that are used by the English Morphological Analyzer are written as capitalized, whereas the super-categories that are added to arrange the lattice are written in all capitals.

Table C.1: English Lattice Structure.

Category Name	**Parent Category Name**
ANY	
Verb	ANY
Det	ANY
DET-SUF	ANY
DET-SUF-COUNT	DET-SUF
PRON-SUF	ANY
PRON-SUF-CASE	PRON-SUF
PRON-SUF-COUNT	PRON-SUF
Prep	ANY
NOUN-SUF	ANY
NOUN-SUF-COUNT	NOUN-SUF
ADJ-SUF	ANY
VERB-SUF	ANY
VERB-SUF-TENSE	VERB-SUF
VERB-SUF-COUNT	VERB-SUF

Table C.1: English Lattice Structure (Continued).

Category Name	**Parent Category Name**
Pl	NOUN-SUF-COUNT VERB-SUF-COUNT DET-SUF-COUNT
Sg	NOUN-SUF-COUNT VERB-SUF-COUNT DET-SUF-COUNT
SP	NOUN-SUF-COUNT VERB-SUF-COUNT DET-SUF-COUNT
VProg	NOUN-SUF ADJ-SUF
PRON	ANY
Pron	PRON
Pron+Rel	PRON
Pron+Pers	PRON
Pron+Poss	PRON
Pron+Wh	PRON
Part	ANY
Interj	ANY
Inf	VERB-SUF-TENSE
PastPerf	VERB-SUF-TENSE
PastSimp	VERB-SUF-TENSE
Pres	VERB-SUF-TENSE
Prog	VERB-SUF-TENSE
1sg	VERB-SUF-COUNT PRON-SUF-COUNT
2sg	VERB-SUF-COUNT PRON-SUF-COUNT
3sg	VERB-SUF-COUNT PRON-SUF-COUNT
123SP	VERB-SUF-COUNT
Non3sg	VERB-SUF-COUNT
1pl	PRON-SUF-COUNT
2pl	PRON-SUF-COUNT
3pl	PRON-SUF-COUNT
3SP	PRON-SUF-COUNT
Gen	PRON-SUF-CASE

Table C.1: English Lattice Structure (Continued).

Category Name	**Parent Category Name**
Nom	PRON-SUF-CASE
Obl	PRON-SUF-CASE
NomObl	PRON-SUF-CASE
Def	DET-SUF
Indef	DET-SUF
CONJ	ANY
Conj+Coord	CONJ
Conj+Sub	CONJ
Comp	ADJ-SUF
Sup	ADJ-SUF
NUMBER	ANY
Num+Card	NUMBER
Num+Ord	NUMBER
Noun	ANY
NOUN-TYPE	ANY
Prop	NOUN-TYPE
Adj	ANY
Adv	ANY
Aux	ANY
Punc	ANY
NOUN-DB	ANY
ADVERB-DB	ANY
VERB-DB	ANY
ADJ-DB	ANY
NUM-DB	ANY
ADV-DB	ANY
NUM-DB-ADJ	NUM-DB
ADJ-DB-VERB	ADJ-DB
VERB-DB-NOUN	VERB-DB
NOUN-DB-VERB	NOUN-DB
ADV-DB-ADV	ADV-DB
ADJ-DB-ADV	ADJ-DB
VERB-DB-ADJ	VERB-DB
VERB-DB-VERB	VERB-DB
NOUN-DB-ADJ	NOUN-DB
ADJ-DB-NOUN	ADJ-DB

Table C.1: English Lattice Structure (Continued).

Category Name	**Parent Category Name**
ADJ-DB-ADJ	ADJ-DB
NOUN-DB-NOUN	NOUN-DB
ˆDB+Noun+Sion	VERB-DB-NOUN
ˆDB+Adj+Ive	NOUN-DB-ADJ VERB-DB-ADJ
ˆDB+Noun+Ant	VERB-DB-NOUN
ˆDB+Adj+Ary	NOUN-DB-ADJ NUM-DB-ADJ VERB-DB-ADJ
ˆDB+Adj+Ar	NOUN-DB-ADJ
ˆDB+Noun+Ance	ADJ-DB-NOUN
ˆDB+Noun+Ness	ADJ-DB-NOUN
ˆDB+Noun+Ity	ADJ-DB-NOUN
ˆDB+Adj+Al	NOUN-DB-ADJ ADJ-DB-ADJ
ˆDB+Adj+Itive	NOUN-DB-ADJ VERB-DB-ADJ
ˆDB+Noun+Cy	ADJ-DB-NOUN NOUN-DB-NOUN VERB-DB-NOUN
ˆDB+Noun+Ery	NOUN-DB-NOUN VERB-DB-NOUN ADJ-DB-NOUN
ˆDB+Noun+Ful	NOUN-DB-NOUN
ˆDB+Noun+Ment	VERB-DB-NOUN
ˆDB+Noun+Ite	ADJ-DB-NOUN NOUN-DB-NOUN
ˆDB+Adv+Ly	ADJ-DB-ADV ADV-DB-ADV
ˆDB+Noun+Or	VERB-DB-NOUN
ˆDB+Adj+Ative	NOUN-DB-ADJ VERB-DB-ADJ
ˆDB+Noun+Zero	VERB-DB-NOUN ADJ-DB-NOUN
ˆDB+Adj+Ly	ADJ-DB-ADJ NOUN-DB-ADJ
ˆDB+Adj+Ant	VERB-DB-ADJ

Table C.1: English Lattice Structure (Continued).

Category Name	**Parent Category Name**
ˆDB+Adj+Less	NOUN-DB-ADJ VERB-DB-ADJ
ˆDB+Adj+Ory	NOUN-DB-ADJ VERB-DB-ADJ
ˆDB+Noun+Tion	VERB-DB-NOUN
ˆDB+Noun+Age	NOUN-DB-NOUN VERB-DB-NOUN ADJ-DB-NOUN
ˆDB+Noun+Ist	NOUN-DB-NOUN
ˆDB+Adj+Ous	ADJ-DB-ADJ NOUN-DB-ADJ VERB-DB-ADJ
ˆDB+Adj+Zero	VERB-DB-ADJ NOUN-DB-ADJ
ˆDB+Noun+Ism	NOUN-DB-NOUN
ˆDB+Noun+Ling	NOUN-DB-NOUN ADJ-DB-NOUN VERB-DB-NOUN
ˆDB+Verb+Ate	ADJ-DB-VERB NOUN-DB-VERB VERB-DB-VERB
ˆDB+Noun+Ure	VERB-DB-NOUN ADJ-DB-NOUN NOUN-DB-NOUN
ˆDB+Noun+Ent	VERB-DB-NOUN
ˆDB+Noun+Er	VERB-DB-NOUN NOUN-DB-NOUN ADJ-DB-NOUN
ˆDB+Adj+Ic	NOUN-DB-ADJ
ˆDB+Adj+Ate	NOUN-DB-ADJ VERB-DB-ADJ
ˆDB+Noun+Ty	ADJ-DB-NOUN
ˆDB+Adj+Ful	NOUN-DB-ADJ VERB-DB-ADJ ADJ-DB-ADJ

Table C.1: English Lattice Structure (Continued).

Category Name	**Parent Category Name**
ˆDB+Noun+Ary	NOUN-DB-NOUN VERB-DB-NOUN ADJ-DB-NOUN
ˆDB+Verb+Ize	NOUN-DB-VERB ADJ-DB-VERB
ˆDB+Adj+Able	VERB-DB-ADJ NOUN-DB-ADJ
ˆDB+Noun+Let	NOUN-DB-NOUN
ˆDB+Noun+Ence	ADJ-DB-NOUN
ˆDB+Noun+Ee	VERB-DB-NOUN
ˆDB+Adj+Y	VERB-DB-ADJ NOUN-DB-ADJ
ˆDB+Adv+Zero	ADJ-DB-ADV
ˆDB+Adj+Atory	NOUN-DB-ADJ VERB-DB-ADJ
ˆDB+Noun+Th	ADJ-DB-NOUN
ˆDB+Adj+En	NOUN-DB-ADJ VERB-DB-ADJ
ˆDB+Verb+Zero	NOUN-DB-VERB ADJ-DB-VERB
ˆDB+Noun+Ion	VERB-DB-NOUN
ˆDB+Adj+Ed	NOUN-DB-ADJ
ˆDB+Adj+Ent	VERB-DB-ADJ
ˆDB+Noun+Al	VERB-DB-NOUN
ˆDB+Noun+Y	VERB-DB-NOUN
ˆDB+Noun+Ess	NOUN-DB-NOUN
ˆDB+Noun+Eer	NOUN-DB-NOUN
ˆDB+Verb+Ise	NOUN-DB-VERB ADJ-DB-VERB
ˆDB+Adj+Ible	VERB-DB-ADJ NOUN-DB-ADJ
ˆDB+Noun+Ation	VERB-DB-NOUN
ˆDB+Noun+Ster	NOUN-DB-NOUN
ˆDB+Adj+Ish	NOUN-DB-ADJ VERB-DB-ADJ
ˆDB+Verb+Ify	ADJ-DB-VERB NOUN-DB-VERB

Appendix D

Evaluation Data Set

This appendix lists the translation examples that were used in the performance evaluation of the system (see Chapter 6). The data set is divided into 3 subsets. Training Subset 1 contains the examples from which the translation templates were extracted. Translation examples in Training Subset 2 were used for teaching the system some context-dependent co-occurrence rules. Lastly, Testing Subset contains, as the name implies, the examples that were used for the testing purposes. The following sections list the examples in each of these subsets.

D.1 Training Subset 1

1. a+Det +Indef +Sg brown+Adj car+Noun +Sg ↔ bir+Num+Card kahverengi+Adj araba+Noun +A3sg +Pnon +Nom
2. a+Det +Indef +Sg cat+Noun +Sg come+Verb +Pres +3sg ↔ bir+Num+Card kedi+Noun +A3sg +Pnon +Nom gel+Verb +Pos +Aor +A3sg
3. a+Det +Indef +Sg cat+Noun +Sg go+Verb +Pres +3sg ↔ bir+Num+Card kedi+Noun +A3sg +Pnon +Nom git+Verb +Pos +Aor +A3sg
4. a+Det +Indef +Sg green+Adj apple+Noun +Sg ↔ bir+Num+Card yeşil+Adj elma+Noun +A3sg +Pnon +Nom
5. a+Det +Indef +Sg pig+Noun +Sg go+Verb +Pres +3sg ↔ bir+Num+Card domuz+Noun +A3sg +Pnon +Nom git+Verb +Pos +Aor +A3sg
6. a+Det +Indef +Sg yellow+Adj apple+Noun +Sg ↔ bir+Num+Card sarı+Adj elma+Noun +A3sg +Pnon +Nom
7. a+Det +Indef +Sg white+Adj car+Noun +Sg ↔ bir+Num+Card beyaz+Adj araba+Noun +A3sg +Pnon +Nom

8. a+Det +Indef +Sg yellow+Adj apple+Noun +Sg ↔ bir+Num+Card sarı+Adj elma+Noun +A3sg +Pnon +Nom
9. the+Det +Def +SP yellow+Adj apple+Noun +Sg ↔ sarı+Adj elma+Noun +A3sg +Pnon +Nom
10. the+Det +Def +SP green+Adj apple+Noun +Sg ↔ yeşil+Adj elma+Noun +A3sg +Pnon +Nom
11. the+Det +Def +SP white+Adj car+Noun +Sg ↔ beyaz+Adj araba+Noun +A3sg +Pnon +Nom
12. the+Det +Def +SP white+Adj car+Noun +Pl ↔ beyaz+Adj araba+Noun +A3pl +Pnon +Nom
13. the+Det +Def +SP yellow+Adj apple+Noun +Sg ↔ sarı+Adj elma+Noun +A3sg +Pnon +Nom
14. the+Det +Def +SP yellow+Adj apple+Noun +Pl ↔ sarı+Adj elma+Noun +A3pl +Pnon +Nom
15. black+Adj book+Noun +Sg ↔ siyah+Adj kitap+Noun +A3sg +Pnon +Nom
16. black+Adj car+Noun +Sg ↔ siyah+Adj araba+Noun +A3sg +Pnon +Nom
17. black+Adj notebook+Noun +Sg ↔ siyah+Adj defter+Noun +A3sg +Pnon +Nom
18. blue+Adj book+Noun +Sg ↔ mavi+Adj kitap+Noun +A3sg +Pnon +Nom
19. blue+Adj notebook+Noun +Sg ↔ mavi+Adj defter+Noun +A3sg +Pnon +Nom
20. all+Det +Pl book+Noun +Pl ↔ bütün+Adj kitap+Noun +A3pl +Pnon +Nom
21. all+Det +Pl house+Noun +Pl ↔ bütün+Adj ev+Noun +A3pl +Pnon +Nom
22. all+Det +Pl notebook+Noun +Pl ↔ bütün+Adj defter+Noun +A3pl +Pnon +Nom
23. every+Det +Sg book+Noun +Sg ↔ her+Adj kitap+Noun +A3sg +Pnon +Nom
24. every+Det +Sg house+Noun +Sg ↔ her+Adj ev+Noun +A3sg +Pnon +Nom
25. every+Det +Sg notebook+Noun +Sg ↔ her+Adj defter+Noun +A3sg +Pnon +Nom
26. every+Det +Sg school+Noun +Sg ↔ her+Adj okul+Noun +A3sg +Pnon +Nom
27. one+Num+Ord house+Noun +Sg ↔ bir+Num+Ord ev+Noun +A3sg +Pnon +Nom
28. one+Num+Ord notebook+Noun +Sg ↔ bir+Num+Ord defter+Noun +A3sg +Pnon +Nom
29. one+Num+Ord school+Noun +Sg ↔ bir+Num+Ord okul+Noun +A3sg +Pnon +Nom
30. at+Prep least+Adv ↔ en+Adverb +AdjMdfy az+Adj
31. at+Prep least+Adv one+Num+Card book+Noun +Sg ↔ en+Adverb +AdjMdfy az+Adverb +AdjMdfy bir+Num+Card kitap+Noun +A3sg +Pnon +Nom
32. at+Prep least+Adv one+Num+Card notebook+Noun +Sg ↔ en+Adverb +AdjMdfy az+Adverb +AdjMdfy bir+Num+Card defter+Noun +A3sg +Pnon +Nom
33. at+Prep least+Adv three+Num+Card book+Noun +Pl ↔ en+Adverb +AdjMdfy az+Adverb +AdjMdfy üç+Num+Card kitap+Noun +A3sg +Pnon +Nom
34. at+Prep least+Adv two+Num+Card book+Noun +Pl ↔ en+Adverb +AdjMdfy az+Adverb +AdjMdfy iki+Num+Card kitap+Noun +A3sg +Pnon +Nom
35. at+Prep most+Adv ↔ en+Adverb +AdjMdfy çok+Adverb +AdjMdfy
36. at+Prep most+Adv one+Num+Card book+Noun +Sg ↔ en+Adverb +AdjMdfy çok+Adverb +AdjMdfy bir+Num+Card kitap+Noun +A3sg +Pnon +Nom
37. at+Prep most+Adv three+Num+Card book+Noun +Pl ↔ en+Adverb +AdjMdfy çok+Adverb +AdjMdfy üç+Num+Card kitap+Noun +A3sg +Pnon +Nom
38. at+Prep most+Adv three+Num+Card notebook+Noun +Pl ↔ en+Adverb +AdjMdfy çok+Adverb +AdjMdfy üç+Num+Card defter+Noun +A3sg +Pnon +Nom
39. at+Prep most+Adv two+Num+Card book+Noun +Pl ↔ en+Adverb +AdjMdfy çok+Adverb +AdjMdfy iki+Num+Card kitap+Noun +A3sg +Pnon +Nom

40. ali+Noun +Prop +Sg +Part +Gen notebook+Noun +Sg ↔ ali+Noun +Prop +A3sg +Pnon +Gen defter+Noun +A3sg +P3sg +Nom
41. ali+Noun +Prop +Sg +Part +Gen school+Noun +Sg ↔ ali+Noun +Prop +A3sg +Pnon +Gen okul+Noun +A3sg +P3sg +Nom
42. four+Num+Card black+Adj car+Noun +Pl ↔ dört+Num+Card siyah+Adj araba+Noun +A3sg +Pnon +Nom
43. four+Num+Card green+Adj apple+Noun +Pl ↔ dört+Num+Card yeşil+Adj elma+Noun +A3sg +Pnon +Nom
44. four+Num+Card yellow+Adj apple+Noun +Pl ↔ dört+Num+Card sarı+Adj elma+Noun +A3sg +Pnon +Nom
45. four+Num+Card white+Adj car+Noun +Pl ↔ dört+Num+Card beyaz+Adj araba+Noun +A3sg +Pnon +Nom
46. four+Num+Card yellow+Adj apple+Noun +Pl ↔ dört+Num+Card sarı+Adj elma+Noun +A3sg +Pnon +Nom
47. boy+Noun +Pl be+Verb +Pres +Pl come+Verb +Prog ↔ oğlan+Noun +A3pl +Pnon +Nom gel+Verb +Pos +Prog1 +A3pl
48. boy+Noun +Pl be+Verb +Pres +Pl not+Adv come+Verb +Prog ↔ oğlan+Noun +A3pl +Pnon +Nom gel+Verb +Neg +Prog1 +A3pl
49. boy+Noun +Pl be+Verb +Pres +Pl not+Adv go+Verb +Prog ↔ oğlan+Noun +A3pl +Pnon +Nom git+Verb +Neg +Prog1 +A3pl
50. girl+Noun +Pl be+Verb +Pres +Pl go+Verb +Prog ↔ kız+Noun +A3pl +Pnon +Nom git+Verb +Pos +Prog1 +A3pl
51. girl+Noun +Pl be+Verb +Pres +Pl not+Adv come+Verb +Prog ↔ kız+Noun +A3pl +Pnon +Nom gel+Verb +Neg +Prog1 +A3pl
52. girl+Noun +Pl be+Verb +Pres +Pl not+Adv go+Verb +Prog ↔ kız+Noun +A3pl +Pnon +Nom git+Verb +Neg +Prog1 +A3pl
53. thief+Noun +Sg ↔ hırsız+Noun +A3sg +Pnon +Nom
54. cop+Noun +Sg ↔ polis+Noun +A3sg +Pnon +Nom
55. to+Prep steal+Verb +Inf ↔ çal+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
56. to+Prep approach+Verb +Inf ↔ yaklaş+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
57. girl+Noun +Pl will+Aux come+Verb +Pres +Non3sg ↔ kız+Noun +A3pl +Pnon +Nom gel+Verb +Pos +Fut +A3pl
58. girl+Noun +Pl will+Aux go+Verb +Pres +Non3sg ↔ kız+Noun +A3pl +Pnon +Nom git+Verb +Pos +Fut +A3pl
59. girl+Noun +Pl will+Aux go+Verb +Pres +Non3sg to+Prep the+Det +Def +SP mountain+Noun +Sg ↔ kız+Noun +A3pl +Pnon +Nom dağ+Noun +A3sg +Pnon +Dat git+Verb +Pos +Fut +A3pl
60. girl+Noun +Pl will+Aux go+Verb +Pres +Non3sg to+Prep the+Det +Def +SP mountain+Noun +Sg tomorrow+Adv ↔ kız+Noun +A3pl +Pnon +Nom yarın+Adverb dağ+Noun +A3sg +Pnon +Dat git+Verb +Pos +Fut +A3pl
61. girl+Noun +Pl will+Aux go+Verb +Pres +Non3sg tomorrow+Adv ↔ kız+Noun +A3pl +Pnon +Nom yarın+Adverb git+Verb +Pos +Fut +A3pl
62. girl+Noun +Pl will+Aux not+Adv come+Verb +Pres +Non3sg tomorrow+Adv ↔ kız+Noun +A3pl +Pnon +Nom yarın+Adverb gel+Verb +Neg +Fut +A3pl
63. girl+Noun +Pl will+Aux not+Adv go+Verb +Pres +Non3sg ↔ kız+Noun +A3pl +Pnon +Nom git+Verb +Neg +Fut +A3pl
64. girl+Noun +Pl will+Aux not+Adv go+Verb +Pres +Non3sg to+Prep the+Det +Def +SP mountain+Noun +Sg ↔ kız+Noun +A3pl +Pnon +Nom dağ+Noun +A3sg +Pnon +Dat git+Verb +Neg +Fut +A3pl

65. girl+Noun +Pl will+Aux not+Adv go+Verb +Pres +Non3sg to+Prep the+Det +Def +SP mountain+Noun +Sg tomorrow+Adv ↔ kız+Noun +A3pl +Pnon +Nom yarın+Adverb dağ+Noun +A3sg +Pnon +Dat git+Verb +Neg +Fut +A3pl
66. girl+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg a+Det +Indef +Sg letter+Noun +Sg ↔ kız+Noun +A3pl +Pnon +Nom bir+Num+Card mektup+Noun +A3sg +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
67. girl+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg a+Det +Indef +Sg letter+Noun +Sg tomorrow+Adv ↔ kız+Noun +A3pl +Pnon +Nom yarın+Adverb bir+Num+Card mektup+Noun +A3sg +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
68. girl+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg a+Det +Indef +Sg message+Noun +Sg ↔ kız+Noun +A3pl +Pnon +Nom bir+Num+Card mesaj+Noun +A3sg +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
69. girl+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg a+Det +Indef +Sg message+Noun +Sg tomorrow+Adv ↔ kız+Noun +A3pl +Pnon +Nom yarın+Adverb bir+Num+Card mesaj+Noun +A3sg +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
70. girl+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg letter+Noun +Pl tomorrow+Adv ↔ kız+Noun +A3pl +Pnon +Nom yarın+Adverb mektup+Noun +A3pl +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
71. girl+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg message+Noun +Pl ↔ kız+Noun +A3pl +Pnon +Nom mesaj+Noun +A3pl +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
72. girl+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg message+Noun +Pl tomorrow+Adv ↔ kız+Noun +A3pl +Pnon +Nom yarın+Adverb mesaj+Noun +A3pl +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
73. girl+Noun +Pl will+Aux write+Verb +Pres +Non3sg a+Det +Indef +Sg letter+Noun +Sg ↔ kız+Noun +A3pl +Pnon +Nom bir+Num+Card mektup+Noun +A3sg +Pnon +Nom yaz+Verb +Pos +Fut +A3pl
74. girl+Noun +Pl will+Aux write+Verb +Pres +Non3sg a+Det +Indef +Sg message+Noun +Sg ↔ kız+Noun +A3pl +Pnon +Nom bir+Num+Card mesaj+Noun +A3sg +Pnon +Nom yaz+Verb +Pos +Fut +A3pl
75. girl+Noun +Pl will+Aux write+Verb +Pres +Non3sg a+Det +Indef +Sg message+Noun +Sg tomorrow+Adv ↔ kız+Noun +A3pl +Pnon +Nom yarın+Adverb bir+Num+Card mesaj+Noun +A3sg +Pnon +Nom yaz+Verb +Pos +Fut +A3pl
76. girl+Noun +Pl will+Aux write+Verb +Pres +Non3sg letter+Noun +Pl ↔ kız+Noun +A3pl +Pnon +Nom mektup+Noun +A3pl +Pnon +Nom yaz+Verb +Pos +Fut +A3pl
77. girl+Noun +Pl will+Aux write+Verb +Pres +Non3sg letter+Noun +Pl tomorrow+Adv ↔ kız+Noun +A3pl +Pnon +Nom yarın+Adverb mektup+Noun +A3pl +Pnon +Nom yaz+Verb +Pos +Fut +A3pl
78. girl+Noun +Pl will+Aux write+Verb +Pres +Non3sg message+Noun +Pl tomorrow+Adv ↔ kız+Noun +A3pl +Pnon +Nom yarın+Adverb mesaj+Noun +A3pl +Pnon +Nom yaz+Verb +Pos +Fut +A3pl
79. boy+Noun +Pl will+Aux come+Verb +Pres +Non3sg ↔ oğlan+Noun +A3pl +Pnon +Nom gel+Verb +Pos +Fut +A3pl
80. boy+Noun +Pl will+Aux go+Verb +Pres +Non3sg ↔ oğlan+Noun +A3pl +Pnon +Nom git+Verb +Pos +Fut +A3pl
81. boy+Noun +Pl will+Aux go+Verb +Pres +Non3sg to+Prep the+Det +Def +SP mountain+Noun +Sg ↔ oğlan+Noun +A3pl +Pnon +Nom dağ+Noun +A3sg +Pnon +Dat git+Verb +Pos +Fut +A3pl
82. boy+Noun +Pl will+Aux go+Verb +Pres +Non3sg to+Prep the+Det +Def +SP mountain+Noun +Sg tomorrow+Adv ↔ oğlan+Noun +A3pl +Pnon +Nom yarın+Adverb dağ+Noun +A3sg +Pnon +Dat git+Verb +Pos +Fut +A3pl

83. boy+Noun +Pl will+Aux go+Verb +Pres +Non3sg tomorrow+Adv ↔ oğlan+Noun +A3pl +Pnon +Nom yarın+Adverb git+Verb +Pos +Fut +A3pl
84. boy+Noun +Pl will+Aux not+Adv come+Verb +Pres +Non3sg tomorrow+Adv ↔ oğlan+Noun +A3pl +Pnon +Nom yarın+Adverb gel+Verb +Neg +Fut +A3pl
85. boy+Noun +Pl will+Aux not+Adv go+Verb +Pres +Non3sg ↔ oğlan+Noun +A3pl +Pnon +Nom git+Verb +Neg +Fut +A3pl
86. boy+Noun +Pl will+Aux not+Adv go+Verb +Pres +Non3sg to+Prep the+Det +Def +SP mountain+Noun +Sg ↔ oğlan+Noun +A3pl +Pnon +Nom dağ+Noun +A3sg +Pnon +Dat git+Verb +Neg +Fut +A3pl
87. boy+Noun +Pl will+Aux not+Adv go+Verb +Pres +Non3sg to+Prep the+Det +Def +SP mountain+Noun +Sg tomorrow+Adv ↔ oğlan+Noun +A3pl +Pnon +Nom yarın+Adverb dağ+Noun +A3sg +Pnon +Dat git+Verb +Neg +Fut +A3pl
88. boy+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg a+Det +Indef +Sg letter+Noun +Sg ↔ oğlan+Noun +A3pl +Pnon +Nom bir+Num+Card mektup+Noun +A3sg +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
89. boy+Noun +Pl will+Aux not+Adv read+Verb +Pres +Non3sg a+Det +Indef +Sg letter+Noun +Sg ↔ oğlan+Noun +A3pl +Pnon +Nom bir+Num+Card mektup+Noun +A3sg +Pnon +Nom oku+Verb +Neg +Fut +A3pl
90. boy+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg a+Det +Indef +Sg letter+Noun +Sg tomorrow+Adv ↔ oğlan+Noun +A3pl +Pnon +Nom yarın+Adverb bir+Num+Card mektup+Noun +A3sg +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
91. boy+Noun +Pl will+Aux not+Adv read+Verb +Pres +Non3sg a+Det +Indef +Sg letter+Noun +Sg tomorrow+Adv ↔ oğlan+Noun +A3pl +Pnon +Nom yarın+Adverb bir+Num+Card mektup+Noun +A3sg +Pnon +Nom oku+Verb +Neg +Fut +A3pl
92. boy+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg a+Det +Indef +Sg message+Noun +Sg ↔ oğlan+Noun +A3pl +Pnon +Nom bir+Num+Card mesaj+Noun +A3sg +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
93. boy+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg a+Det +Indef +Sg message+Noun +Sg tomorrow+Adv ↔ oğlan+Noun +A3pl +Pnon +Nom yarın+Adverb bir+Num+Card mesaj+Noun +A3sg +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
94. boy+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg letter+Noun +Pl tomorrow+Adv ↔ oğlan+Noun +A3pl +Pnon +Nom yarın+Adverb mektup+Noun +A3pl +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
95. boy+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg message+Noun +Pl ↔ oğlan+Noun +A3pl +Pnon +Nom mesaj+Noun +A3pl +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
96. boy+Noun +Pl will+Aux not+Adv read+Verb +Pres +Non3sg message+Noun +Pl ↔ oğlan+Noun +A3pl +Pnon +Nom mesaj+Noun +A3pl +Pnon +Nom oku+Verb +Neg +Fut +A3pl
97. boy+Noun +Pl will+Aux not+Adv write+Verb +Pres +Non3sg message+Noun +Pl tomorrow+Adv ↔ oğlan+Noun +A3pl +Pnon +Nom yarın+Adverb mesaj+Noun +A3pl +Pnon +Nom yaz+Verb +Neg +Fut +A3pl
98. boy+Noun +Pl will+Aux not+Adv read+Verb +Pres +Non3sg message+Noun +Pl tomorrow+Adv ↔ oğlan+Noun +A3pl +Pnon +Nom yarın+Adverb mesaj+Noun +A3pl +Pnon +Nom okus+Verb +Neg +Fut +A3pl
99. boy+Noun +Pl will+Aux write+Verb +Pres +Non3sg a+Det +Indef +Sg letter+Noun +Sg ↔ oğlan+Noun +A3pl +Pnon +Nom bir+Num+Card mektup+Noun +A3sg +Pnon +Nom yaz+Verb +Pos +Fut +A3pl

100. boy+Noun +Pl will+Aux write+Verb +Pres +Non3sg a+Det +Indef +Sg message+Noun +Sg ↔ oğlan+Noun +A3pl +Pnon +Nom bir+Num+Card mesaj+Noun +A3sg +Pnon +Nom yaz+Verb +Pos +Fut +A3pl
101. boy+Noun +Pl will+Aux write+Verb +Pres +Non3sg a+Det +Indef +Sg message+Noun +Sg tomorrow+Adv ↔ oğlan+Noun +A3pl +Pnon +Nom yarın+Adverb bir+Num+Card mesaj+Noun +A3sg +Pnon +Nom yaz+Verb +Pos +Fut +A3pl
102. boy+Noun +Pl will+Aux write+Verb +Pres +Non3sg letter+Noun +Pl ↔ oğlan+Noun +A3pl +Pnon +Nom mektup+Noun +A3pl +Pnon +Nom yaz+Verb +Pos +Fut +A3pl
103. boy+Noun +Pl will+Aux read+Verb +Pres +Non3sg letter+Noun +Pl ↔ oğlan+Noun +A3pl +Pnon +Nom mektup+Noun +A3pl +Pnon +Nom oku+Verb +Pos +Fut +A3pl
104. boy+Noun +Pl will+Aux write+Verb +Pres +Non3sg book+Noun +Pl ↔ oğlan+Noun +A3pl +Pnon +Nom kitap+Noun +A3pl +Pnon +Nom yaz+Verb +Pos +Fut +A3pl
105. boy+Noun +Pl will+Aux write+Verb +Pres +Non3sg letter+Noun +Pl tomorrow+Adv ↔ oğlan+Noun +A3pl +Pnon +Nom yarın+Adverb mektup+Noun +A3pl +Pnon +Nom yaz+Verb +Pos +Fut +A3pl
106. boy+Noun +Pl will+Aux write+Verb +Pres +Non3sg message+Noun +Pl tomorrow+Adv ↔ oğlan+Noun +A3pl +Pnon +Nom yarın+Adverb mesaj+Noun +A3pl +Pnon +Nom yaz+Verb +Pos +Fut +A3pl
107. this+Det +Sg school+Noun +Sg ↔ bu+Adj okul+Noun +A3sg +Pnon +Nom
108. these+Det +Pl book+Noun +Pl ↔ bu+Adj kitap+Noun +A3pl +Pnon +Nom
109. these+Det +Pl house+Noun +Pl ↔ bu+Adj ev+Noun +A3pl +Pnon +Nom
110. these+Det +Pl school+Noun +Pl ↔ bu+Adj okul+Noun +A3pl +Pnon +Nom
111. those+Det +Pl book+Noun +Pl ↔ şu+Adj kitap+Noun +A3pl +Pnon +Nom
112. those+Det +Pl house+Noun +Pl ↔ şu+Adj ev+Noun +A3pl +Pnon +Nom
113. those+Det +Pl notebook+Noun +Pl ↔ şu+Adj defter+Noun +A3pl +Pnon +Nom
114. three+Num+Card book+Noun +Pl ↔ üç+Num+Card kitap+Noun +A3sg +Pnon +Nom
115. three+Num+Card house+Noun +Pl ↔ üç+Num+Card ev+Noun +A3sg +Pnon +Nom
116. three+Num+Card notebook+Noun +Pl ↔ üç+Num+Card defter+Noun +A3sg +Pnon +Nom
117. three+Num+Card school+Noun +Pl ↔ üç+Num+Card okul+Noun +A3sg +Pnon +Nom
118. two+Num+Card book+Noun +Pl ↔ iki+Num+Card kitap+Noun +A3sg +Pnon +Nom
119. two+Num+Card brown+Adj car+Noun +Pl ↔ iki+Num+Card kahverengi+Adj araba+Noun +A3sg +Pnon +Nom
120. two+Num+Card green+Adj apple+Noun +Pl ↔ iki+Num+Card yeşil+Adj elma+Noun +A3sg +Pnon +Nom
121. two+Num+Card house+Noun +Pl ↔ iki+Num+Card ev+Noun +A3sg +Pnon +Nom
122. two+Num+Card yellow+Adj apple+Noun +Pl ↔ iki+Num+Card sarı+Adj elma+Noun +A3sg +Pnon +Nom
123. two+Num+Card school+Noun +Pl ↔ iki+Num+Card okul+Noun +A3sg +Pnon +Nom
124. two+Num+Card white+Adj car+Noun +Pl ↔ iki+Num+Card beyaz+Adj araba+Noun +A3sg +Pnon +Nom
125. two+Num+Card yellow+Adj apple+Noun +Pl ↔ iki+Num+Card sarı+Adj elma+Noun +A3sg +Pnon +Nom
126. cold+Adj ↔ soğuk+Adj
127. whole+Adj ↔ bütün+Adj

128. heavy+Adj ↔ ağır+Adj
129. heavy+Adj ↔ zor+Adj
130. hard+Adj ↔ zor+Adj
131. hard+Adj ↔ sert+Adj
132. difficult+Adj ↔ zor+Adj
133. mouth+Noun +Sg ↔ ağız+Noun +A3sg +Pnon +Nom
134. rim+Noun +Sg ↔ ağız+Noun +A3sg +Pnon +Nom
135. brim+Noun +Sg ↔ ağız+Noun +A3sg +Pnon +Nom
136. beer+Noun +Sg ↔ bira+Noun +A3sg +Pnon +Nom
137. whiskey+Noun +Sg ↔ viski+Noun +A3sg +Pnon +Nom
138. week+Noun +Sg ↔ hafta+Noun +A3sg +Pnon +Nom
139. you+Pron+Pers +Nom +2sg be+Verb +Pres +Pl a+Det +Indef +Sg tailor+Noun +Sg ↔ sen+Pron +A2sg +Pnon +Nom bir+Num+Card terzi+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A2sg
140. you+Pron+Pers +Nom +2sg be+Verb +Pres +Pl a+Det +Indef +Sg cop+Noun +Sg ↔ sen+Pron +A2sg +Pnon +Nom bir+Num+Card polis+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A2sg
141. i+Pron+Pers +Nom +1sg be+Verb +Pres +1sg a+Det +Indef +Sg tailor+Noun +Sg ↔ ben+Pron +A1sg +Pnon +Nom bir+Num+Card terzi+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A1sg
142. i+Pron+Pers +Nom +1sg be+Verb +Pres +1sg a+Det +Indef +Sg cop+Noun +Sg ↔ ben+Pron +A1sg +Pnon +Nom bir+Num+Card polis+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A1sg
143. it+Pron+Pers +Nom +3sg be+Verb +Pres +3sg a+Det +Indef +Sg car+Noun +Sg ↔ o+Pron +A3sg +Pnon +Nom bir+Num+Card araba+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +Cop +A3sg
144. you+Pron+Pers +Nom +2sg be+Verb +Pres +Pl not+Adv a+Det +Indef +Sg tailor+Noun +Sg ↔ sen+Pron +A2sg +Pnon +Nom bir+Num+Card terzi+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A2sg
145. you+Pron+Pers +Nom +2sg be+Verb +Pres +Pl not+Adv a+Det +Indef +Sg cop+Noun +Sg ↔ sen+Pron +A2sg +Pnon +Nom bir+Num+Card polis+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A2sg
146. i+Pron+Pers +Nom +1sg be+Verb +Pres +1sg not+Adv a+Det +Indef +Sg tailor+Noun +Sg ↔ ben+Pron +A1sg +Pnon +Nom bir+Num+Card terzi+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A1sg
147. i+Pron+Pers +Nom +1sg be+Verb +Pres +1sg not+Adv a+Det +Indef +Sg cop+Noun +Sg ↔ ben+Pron +A1sg +Pnon +Nom bir+Num+Card polis+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A1sg
148. it+Pron+Pers +Nom +3sg be+Verb +Pres +3sg not+Adv a+Det +Indef +Sg car+Noun +Sg ↔ o+Pron +A3sg +Pnon +Nom bir+Num+Card araba+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +Cop +A3sg
149. they+Pron+Pers +Nom +3pl be+Verb +Pres +Pl cop+Noun +Pl ↔ o+Pron +A3pl +Pnon +Nom polis+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A3pl
150. they+Pron+Pers +Nom +3pl be+Verb +Pres +Pl tailor+Noun +Pl ↔ o+Pron +A3pl +Pnon +Nom terzi+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A3pl
151. they+Pron+Pers +Nom +3pl be+Verb +Pres +Pl not+Adv cop+Noun +Pl ↔ o+Pron +A3pl +Pnon +Nom polis+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A3pl

152. they+Pron+Pers +Nom +3pl be+Verb +Pres +Pl not+Adv tailor+Noun +Pl ↔ o+Pron +A3pl +Pnon +Nom terzi+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ^DB+Verb+Zero +Pres +A3pl
153. which+Pron+Wh +NomObl +3SP book+Noun +Sg ↔ hangi+Adj +Ques kitap+Noun +A3sg +Pnon +Nom
154. which+Pron+Wh +NomObl +3SP book+Noun +Pl ↔ hangi+Adj +Ques kitap+Noun +A3pl +Pnon +Nom
155. which+Pron+Wh +NomObl +3SP house+Noun +Sg ↔ hangi+Adj +Ques ev+Noun +A3sg +Pnon +Nom
156. which+Pron+Wh +NomObl +3SP house+Noun +Pl ↔ hangi+Adj +Ques ev+Noun +A3pl +Pnon +Nom
157. custom+Noun +Sg ↔ adet+Noun +A3sg +Pnon +Nom
158. habit+Noun +Sg ↔ adet+Noun +A3sg +Pnon +Nom
159. rock+Noun +Sg ↔ kaya+Noun +A3sg +Pnon +Nom
160. problem+Noun +Sg ↔ sorun+Noun +A3sg +Pnon +Nom
161. nature+Noun +Sg ↔ doğa+Noun +A3sg +Pnon +Nom
162. food+Noun +Sg ↔ yiyecek+Noun +A3sg +Pnon +Nom
163. to+Prep take+Verb +Inf ↔ al+Verb +Pos ^DB+Noun+Inf1 +A3sg +Pnon +Nom
164. to+Prep get+Verb +Inf ↔ al+Verb +Pos ^DB+Noun+Inf1 +A3sg +Pnon +Nom
165. to+Prep buy+Verb +Inf ↔ al+Verb +Pos ^DB+Noun+Inf1 +A3sg +Pnon +Nom
166. to+Prep call+Verb +Inf ↔ ara+Verb +Pos ^DB+Noun+Inf1 +A3sg +Pnon +Nom
167. to+Prep call+Verb +Inf ↔ çağır+Verb +Pos ^DB+Noun+Inf1 +A3sg +Pnon +Nom
168. take+Verb +Pres +Non3sg this+Pron +NomObl +3sg ↔ bu+Pron +A3sg +Pnon +Acc al+Verb +Pos +Imp +A2sg
169. heavy+Adj ↔ ağır+Adj
170. heavy+Adj ↔ zor+Adj
171. difficult+Adj ↔ zor+Adj
172. strange+Adj ↔ garip+Adj
173. poor+Adj ↔ garip+Adj
174. small+Adj ↔ küçük+Adj
175. tiny+Adj ↔ küçük+Adj
176. mother+Noun +Sg ↔ anne+Noun +A3sg +Pnon +Nom
177. mother+Noun +Sg ↔ ana+Noun +A3sg +Pnon +Nom
178. mom+Noun +Sg ↔ anne+Noun +A3sg +Pnon +Nom
179. space+Noun +Sg ↔ aralık+Noun +A3sg +Pnon +Nom
180. space+Noun +Sg ↔ uzay+Noun +A3sg +Pnon +Nom
181. paper+Noun +Sg ↔ kağıt+Noun +A3sg +Pnon +Nom
182. paper+Noun +Sg ↔ makale+Noun +A3sg +Pnon +Nom
183. gap+Noun +Sg ↔ aralık+Noun +A3sg +Pnon +Nom
184. december+Noun +Sg ↔ aralık+Noun +A3sg +Pnon +Nom
185. interval+Noun +Sg ↔ aralık+Noun +A3sg +Pnon +Nom
186. portion+Noun +Sg ↔ bölüm+Noun +A3sg +Pnon +Nom
187. department+Noun +Sg ↔ bölüm+Noun +A3sg +Pnon +Nom
188. slice+Noun +Sg ↔ bölüm+Noun +A3sg +Pnon +Nom
189. chapter+Noun +Sg ↔ bölüm+Noun +A3sg +Pnon +Nom
190. water+Noun +Sg ↔ su+Noun +A3sg +Pnon +Nom
191. juice+Noun +Sg ↔ su+Noun +A3sg +Pnon +Nom

192. cat+Noun +Sg ↔ kedi+Noun +A3sg +Pnon +Nom
193. girl+Noun +Sg ↔ kız+Noun +A3sg +Pnon +Nom
194. short+Adj tail+Noun +Sg ˆDB+Adj+Ed horse+Noun +Sg be+Verb +Pres +3sg come+Verb +Prog ↔ kısa+Adj kuyruk+Noun +A3sg +Pnon +Nom ˆDB+Adj+With at+Noun +A3sg +Pnon +Nom gel+Verb +Pos +Prog1 +A3sg
195. short+Adj tail+Noun +Sg ˆDB+Adj+Ed cat+Noun +Sg be+Verb +Pres +3sg come+Verb +Prog ↔ kısa+Adj kuyruk+Noun +A3sg +Pnon +Nom ˆDB+Adj+With kedi+Noun +A3sg +Pnon +Nom gel+Verb +Pos +Prog1 +A3sg
196. high+Adj heel+Noun +Sg ˆDB+Adj+Ed girl+Noun +Sg be+Verb +Pres +3sg come+Verb +Prog ↔ uzun+Adj topuk+Noun +A3sg +Pnon +Nom ˆDB+Adj+With kız+Noun +A3sg +Pnon +Nom gel+Verb +Pos +Prog1 +A3sg
197. high+Adj heel+Noun +Sg ˆDB+Adj+Ed woman+Noun +Sg be+Verb +Pres +3sg come+Verb +Prog ↔ uzun+Adj topuk+Noun +A3sg +Pnon +Nom ˆDB+Adj+With kadın+Noun +A3sg +Pnon +Nom gel+Verb +Pos +Prog1 +A3sg
198. nice+Adj girl+Noun +Sg ↔ hoş+Adj kız+Noun +A3sg +Pnon +Nom
199. red+Adj ↔ kırmızı+Adj
200. red+Adj ↔ kızıl+Adj
201. hair+Noun +Sg ↔ saç+Noun +A3sg +Pnon +Nom
202. short+Adj girl+Noun +Sg ↔ kısa+Adj kız+Noun +A3sg +Pnon +Nom
203. black+Adj horse+Noun +Sg ↔ siyah+Adj at+Noun +A3sg +Pnon +Nom
204. red+Adj flag+Noun +Sg ↔ kırmızı+Adj bayrak+Noun +A3sg +Pnon +Nom
205. black+Adj pencil+Noun +Sg ↔ siyah+Adj kalem+Noun +A3sg +Pnon +Nom
206. red+Adj flag+Noun +Sg ↔ kızıl+Adj bayrak+Noun +A3sg +Pnon +Nom
207. red+Adj pencil+Noun +Sg ↔ kırmızı+Adj kalem+Noun +A3sg +Pnon +Nom
208. red+Adj pencil+Noun +Sg ↔ kızıl+Adj kalem+Noun +A3sg +Pnon +Nom
209. red+Adj line+Noun +Sg ↔ kırmızı+Adj çizgi+Noun +A3sg +Pnon +Nom
210. a+Det +Indef +Sg very+Adv nice+Adj girl+Noun +Sg ↔ çok+Adverb +AdjMdfy hoş+Adj bir+Num+Card kız+Noun +A3sg +Pnon +Nom
211. a+Det +Indef +Sg blond+Adj girl+Noun +Sg ↔ sarışın+Adj bir+Num+Card kız+Noun +A3sg +Pnon +Nom
212. very+Adv nice+Adj ↔ hoş mu hoş+Adj
213. a+Det +Indef +Sg very+Adv nice+Adj lady+Noun +Sg ↔ çok+Adverb +AdjMdfy hoş+Adj bir+Num+Card bayan+Noun +A3sg +Pnon +Nom
214. very+Adv sick+Adj ↔ hasta mı hasta+Adj
215. very+Adv sick+Adj ↔ çok+Adverb +AdjMdfy hasta+Adj
216. literate+Adj ↔ okur yazar+Adj
217. woman+Noun +Sg ↔ kadın+Noun +A3sg +Pnon +Nom
218. man+Noun +Sg ↔ adam+Noun +A3sg +Pnon +Nom
219. free+Adj ↔ boş+Adj
220. empty+Adj ↔ boş+Adj
221. seat+Noun +Sg ↔ koltuk+Noun +A3sg +Pnon +Nom
222. chair+Noun +Sg ↔ sandalye+Noun +A3sg +Pnon +Nom
223. car+Noun +Sg of+Prep the+Det +Def +SP month+Noun +Sg ↔ ay+Noun +A3sg +Pnon +Gen araba+Noun +A3sg +P3sg +Nom
224. monument+Noun +Sg of+Prep the+Det +Def +SP year+Noun +Sg ↔ yıl+Noun +A3sg +Pnon +Gen anıt+Noun +A3sg +P3sg +Nom
225. brim+Noun +Sg of+Prep the+Det +Def +SP cup+Noun +Sg ↔ kupa+Noun +A3sg +Pnon +Gen ağız+Noun +A3sg +P3sg +Nom

226. car+Noun +Sg ↔ araba+Noun +A3sg +Pnon +Nom
227. monument+Noun +Sg ↔ anıt+Noun +A3sg +Pnon +Nom
228. element+Noun +Sg ↔ eleman+Noun +A3sg +Pnon +Nom
229. personnel+Noun +Sg ↔ eleman+Noun +A3sg +Pnon +Nom
230. element+Noun +Sg of+Prep the+Det +Def +SP set+Noun +Sg ↔ küme+Noun +A3sg +Pnon +Gen eleman+Noun +A3sg +P3sg +Nom
231. an+Det +Indef +Sg hour+Noun +Sg ↔ bir+Num+Card saat+Noun +A3sg +Pnon +Nom
232. mary+Noun +Prop +Sg +Part +Gen jacket+Noun +Sg ↔ mary+Noun +Prop +A3sg +Pnon +Gen ceket+Noun +A3sg +P3sg +Nom
233. mary+Noun +Prop +Sg +Part +Gen pencil+Noun +Sg ↔ mary+Noun +Prop +A3sg +Pnon +Gen kalem+Noun +A3sg +P3sg +Nom
234. ahmet+Noun +Prop +Sg +Part +Gen pencil+Noun +Sg ↔ ahmet+Noun +Prop +A3sg +Pnon +Gen kalem+Noun +A3sg +P3sg +Nom
235. tom+Noun +Prop +Sg ↔ tom+Noun +Prop +A3sg +Pnon +Nom
236. john+Noun +Prop +Sg ↔ john+Noun +Prop +A3sg +Pnon +Nom
237. sound+Noun +Sg ↔ ses+Noun +A3sg +Pnon +Nom
238. voice+Noun +Sg ↔ ses+Noun +A3sg +Pnon +Nom
239. sound+Noun +Sg of+Prep the+Det +Def +SP engine+Noun +Sg ↔ motor+Noun +A3sg +Pnon +Gen ses+Noun +A3sg +P3sg +Nom
240. sound+Noun +Sg of+Prep the+Det +Def +SP music+Noun +Sg ↔ müzik+Noun +A3sg +Pnon +Gen ses+Noun +A3sg +P3sg +Nom
241. i+Pron+Pers +Gen +1sg voice+Noun +Sg ↔ ben+Pron +A1sg +Pnon +Gen ses+Noun +A3sg +P1sg +Nom
242. size+Noun +Sg ↔ boy+Noun +A3sg +Pnon +Nom
243. height+Noun +Sg ↔ boy+Noun +A3sg +Pnon +Nom
244. giant+Noun +Sg size+Noun +Sg ↔ battal+Noun +A3sg +Pnon +Nom boy+Noun +A3sg +Pnon +Nom
245. one+Num+Card size+Noun +Sg ↔ tek+Adj boy+Noun +A3sg +Pnon +Nom
246. she+Pron+Pers +Gen +3sg height+Noun +Sg ↔ boy+Noun +A3sg +P3sg +Nom
247. i+Pron+Pers +Gen +1sg size+Noun +Sg ↔ ben+Pron +A1sg +Pnon +Gen beden+Noun +A3sg +P1sg +Nom
248. size+Noun +Sg ↔ beden+Noun +A3sg +Pnon +Nom
249. hour+Noun +Sg ↔ saat+Noun +A3sg +Pnon +Nom
250. watch+Noun +Sg ↔ saat+Noun +A3sg +Pnon +Nom
251. the+Det +Def +SP child+Noun +Sg be+Verb +PastSimp +Sg laugh+Verb +Prog ↔ çocuk+Noun +A3sg +Pnon +Nom gül+Verb +Pos +Prog1 +Past +A3sg
252. the+Det +Def +SP baby+Noun +Sg be+Verb +PastSimp +Sg sleep+Verb +Prog ↔ bebek+Noun +A3sg +Pnon +Nom uyu+Verb +Pos +Prog1 +Past +A3sg
253. to+Prep cry+Verb +Inf ↔ ağla+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
254. to+Prep cry+Verb +Inf ↔ bağır+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
255. to+Prep weep+Verb +Inf ↔ ağla+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
256. do+Aux +Pres +Non3sg not+Adv weep+Verb +Inf at+Prep i+Pron+Pers +Gen +1sg grave+Noun +Sg ↔ mezar+Noun +A3sg +P1sg +Loc ağla+Verb +Pos ˆDB+Noun+Inf2 +A3sg +Pnon +Nom
257. do+Aux +Pres +Non3sg not+Adv cry+Verb +Inf for+Prep i+Pron+Pers +Obl +1sg ↔ ben+Noun +A3sg +P1sg +Nom için+Postp+PCNom ağla+Verb +Neg +Imp +A2sg
258. baby+Noun +Sg ↔ bebek+Noun +A3sg +Pnon +Nom

259. child+Noun +Sg ↔ çocuk+Noun +A3sg +Pnon +Nom
260. animal+Noun +Sg cry+Verb +Inf ˆDB+Noun+Zero +Pl ↔ hayvan+Adj bağır+Verb +Pos ˆDB+Noun+Inf3 +A3pl +P3sg +Nom
261. war+Noun +Sg cry+Verb +Inf ˆDB+Noun+Zero +Sg ↔ savaş+Noun +A3sg +Pnon +Nom bağır+Verb +Pos ˆDB+Noun+Inf3 +A3sg +P3sg +Nom
262. i+Pron+Pers +Gen +1sg plane+Noun +Sg ↔ uçak+Noun +A3sg +P1sg +Nom
263. i+Pron+Pers +Gen +1sg name+Noun +Sg ↔ isim+Noun +A3sg +P1sg +Nom
264. plane+Noun +Sg ↔ düzlem+Noun +A3sg +Pnon +Nom
265. to+Prep crash+Verb +Inf ↔ düş+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
266. to+Prep fly+Verb +Inf ↔ uç+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
267. weep+Verb +Prog ˆDB+Noun+Zero +Sg in+Adv frustrate+Verb +Inf ˆDB+Noun+Ion +Sg ↔ ağla+Verb +Pos ˆDB+Adj+PresPart taş+Adj
268. music+Noun +Sg ↔ müzik+Noun +A3sg +Pnon +Nom
269. to+Prep steal+Verb +Inf ↔ çal+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
270. to+Prep play+Verb +Inf ↔ çal+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
271. to+Prep ring+Verb +Inf ↔ çal+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
272. song+Noun +Sg ↔ şarkı+Noun +A3sg +Pnon +Nom
273. to+Prep insert+Verb +Inf ↔ sok+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
274. to+Prep sting+Verb +Inf ↔ sok+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
275. bee+Noun +Sg ↔ arı+Noun +A3sg +Pnon +Nom
276. scorpion+Noun +Sg ↔ akrep+Noun +A3sg +Pnon +Nom
277. insert+Verb +Inf the+Det +Def +SP coin+Noun +Sg ↔ para+Noun +A3sg +Pnon +Acc sok+Verb +Pos +Imp +A2pl
278. a+Det +Indef +Sg bottle+Noun +Sg of+Prep whiskey+Noun +Sg ↔ bir+Num+Card şişe+Noun +A3sg +Pnon +Nom viski+Noun +A3sg +Pnon +Nom
279. a+Det +Indef +Sg barrel+Noun +Sg of+Prep wine+Noun +Sg ↔ bir+Num+Card fıçı+Noun +A3sg +Pnon +Nom şarap+Noun +A3sg +Pnon +Nom
280. bottle+Noun +Sg ↔ şişe+Noun +A3sg +Pnon +Nom
281. barrel+Noun +Sg ↔ fıçı+Noun +A3sg +Pnon +Nom
282. cup+Noun +Sg ↔ kupa+Noun +A3sg +Pnon +Nom
283. cup+Noun +Sg ↔ fincan+Noun +A3sg +Pnon +Nom
284. world+Noun +Sg cup+Noun +Sg ↔ dünya+Noun +Prop +A3sg +Pnon +Nom kupa+Noun +A3sg +P3sg +Nom
285. mug+Noun +Sg ↔ fincan+Noun +A3sg +Pnon +Nom
286. european+Noun +Sg cup+Noun +Sg ↔ avrupa+Noun +Prop +A3sg +Pnon +Nom kupa+Noun +A3sg +P3sg +Nom
287. coffee+Noun +Sg cup+Noun +Sg ↔ kahve+Noun +A3sg +Pnon +Nom fincan+Noun +A3sg +P3sg +Nom
288. coffee+Noun +Sg ↔ kahve+Noun +A3sg +Pnon +Nom
289. tea+Noun +Sg ↔ çay+Noun +A3sg +Pnon +Nom
290. depart+Verb +Prog ˆDB+Adj+Zero ship+Noun +Sg ↔ git+Verb +Pos ˆDB+Adj+PresPart gemi+Noun +A3sg +Pnon +Nom
291. ship+Noun +Sg ↔ gemi+Noun +A3sg +Pnon +Nom
292. face+Noun +Sg ↔ surat+Noun +A3sg +Pnon +Nom
293. laugh+Verb +Prog ˆDB+Adj+Zero face+Noun +Sg ↔ gül+Verb +Pos ˆDB+Adj+PresPart surat+Noun +A3sg +Pnon +Nom

294. a+Det +Indef +Sg laugh+Verb +Prog ˆDB+Adj+Zero face+Noun +Sg ↔ bir+Num+Card gül+Verb +Pos ˆDB+Adj+PresPart surat+Noun +A3sg +Pnon +Nom
295. a+Det +Indef +Sg blue+Adj carpet+Noun +Sg ↔ bir+Num+Card mavi+Adj halı+Noun +A3sg +Pnon +Nom
296. carpet+Noun +Sg ↔ halı+Noun +A3sg +Pnon +Nom
297. to+Prep grow+Verb +Inf ↔ büyü+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
298. to+Prep rise+Verb +Inf ↔ yüksel+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
299. to+Prep rise+Verb +Inf ↔ doğ+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
300. this+Det +Sg laugh+Verb +Prog ˆDB+Adj+Zero face+Noun +Sg ↔ bu+Pron +A3sg +Pnon +Nom gül+Verb +Pos ˆDB+Adj+PresPart surat+Noun +A3sg +Pnon +Nom
301. this+Det +Sg blue+Adj carpet+Noun +Sg ↔ bu+Pron +A3sg +Pnon +Nom mavi+Adj halı+Noun +A3sg +Pnon +Nom
302. moon+Noun +Sg ↔ ay+Noun +A3sg +Pnon +Nom
303. sun+Noun +Sg ↔ güneş+Noun +A3sg +Pnon +Nom
304. to+Prep rise+Verb +Inf ↔ kalk+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
305. to+Prep glow+Verb +Inf ↔ yan+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
306. to+Prep burn+Verb +Inf ↔ yan+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
307. to+Prep ignite+Verb +Inf ↔ yan+Verb +Pos ˆDB+Noun+Inf1 +A3sg +Pnon +Nom
308. lamp+Noun +Sg ↔ lamba+Noun +A3sg +Pnon +Nom
309. wood+Noun +Sg burn+Verb +PastSimp +123SP ↔ tahta+Adj yan+Verb +Pos +Past +A3sg
310. base+Noun +Sg ↔ ayak+Noun +A3sg +Pnon +Nom
311. foot+Noun +Sg ↔ ayak+Noun +A3sg +Pnon +Nom
312. phone+Noun +Sg ↔ telefon+Noun +A3sg +Pnon +Nom
313. priest+Noun +Sg ↔ rahip+Noun +A3sg +Pnon +Nom
314. bell+Noun +Sg ↔ çan+Noun +A3sg +Pnon +Nom
315. base+Noun +Sg of+Prep the+Det +Def +SP monument+Noun +Sg ↔ anıt+Noun +A3sg +Pnon +Gen ayak+Noun +A3sg +P3sg +Nom

D.2 Training Subset 2

1. red+Adj hair+Noun +Sg ˆDB+Adj+Ed girl+Noun +Sg be+Verb +Pres +3sg come+Verb +Prog ↔ kızıl+Adj saç+Noun +A3sg +Pnon +Nom ˆDB+Adj+With kız+Noun +A3sg +Pnon +Nom gel+Verb +Pos +Prog1 +A3sg
2. a+Det +Indef +Sg very+Adv sick+Adj woman+Noun +Sg ↔ çok+Adverb +AdjMdfy hasta+Adj bir+Num+Card kadın+Noun +A3sg +Pnon +Nom
3. personnel+Noun +Sg of+Prep the+Det +Def +SP month+Noun +Sg ↔ ay+Noun +A3sg +Pnon +Gen eleman+Noun +A3sg +P3sg +Nom
4. john+Noun +Prop +Sg +Part +Gen voice+Noun +Sg ↔ john+Noun +Prop +A3sg +Pnon +Gen ses+Noun +A3sg +P3sg +Nom
5. the+Det +Def +SP child+Noun +Sg be+Verb +PastSimp +Sg cry+Verb +Prog ↔ çocuk+Noun +A3sg +Pnon +Nom ağla+Verb +Pos +Prog1 +Past +A3sg
6. the+Det +Def +SP plane+Noun +Sg be+Verb +PastSimp +Sg crash+Verb +Prog ↔ uçak+Noun +A3sg +Pnon +Nom düş+Verb +Pos +Prog1 +Past +A3sg

7. john+Noun +Prop +Sg +Part +Gen height+Noun +Sg ↔ john+Noun +Prop +A3sg +Pnon +Gen boy+Noun +A3sg +P3sg +Nom
8. a+Det +Indef +Sg cup+Noun +Sg of+Prep tea+Noun +Sg ↔ bir+Num+Card fincan+Noun +A3sg +Pnon +Nom kahve+Noun +A3sg +Pnon +Nom
9. a+Det +Indef +Sg rise+Verb +Prog ˆDB+Adj+Zero sun+Noun +Sg ↔ bir+Num+Card doğ+Verb +Pos ˆDB+Adj+PresPart güneş+Noun +A3sg +Pnon +Nom
10. a+Det +Indef +Sg fly+Verb +Prog ˆDB+Adj+Zero plane+Noun +Sg ↔ bir+Num+Card uç+Verb +Pos ˆDB+Adj+PresPart uçak+Noun +A3sg +Pnon +Nom
11. john+Noun +Prop +Sg +Part +Gen size+Noun +Sg ↔ john+Noun +Prop +A3sg +Pnon +Gen boy+Noun +A3sg +P3sg +Nom
12. a+Det +Indef +Sg empty+Adj seat+Noun +Sg ↔ boş+Adj bir+Num+Card koltuk+Noun +A3sg +Pnon +Nom
13. a+Det +Indef +Sg cry+Verb +Prog ˆDB+Adj+Zero baby+Noun +Sg ↔ bir+Num+Card ağla+Verb +Pos ˆDB+Adj+PresPart bebek+Noun +A3sg +Pnon +Nom
14. the+Det +Def +SP song+Noun +Sg be+Verb +PastSimp +Sg play+Verb +Prog ↔ şarkı+Noun +A3sg +Pnon +Nom çal+Verb +Pos +Prog1 +Past +A3sg
15. john+Noun +Prop +Sg +Part +Gen watch+Noun +Sg ↔ john+Noun +Prop +A3sg +Pnon +Gen saat+Noun +A3sg +P3sg +Nom
16. the+Det +Def +SP bee+Noun +Sg be+Verb +PastSimp +Sg sting+Verb +Prog ↔ arı+Noun +A3sg +Pnon +Nom sok+Verb +Pos +Prog1 +Past +A3sg
17. mother+Noun +Sg of+Prep the+Det +Def +SP child+Noun +Sg ↔ çocuk+Noun +A3sg +Pnon +Gen anne+Noun +A3sg +P3sg +Nom
18. john+Noun +Prop +Sg +Part +Gen mother+Noun +Sg ↔ john+Noun +Prop +A3sg +Pnon +Gen anne+Noun +A3sg +P3sg +Nom
19. a+Det +Indef +Sg month+Noun +Sg ↔ bir+Num+Card ay+Noun +A3sg +Pnon +Nom
20. this+Det +Sg fly+Verb +Prog ˆDB+Adj+Zero plane+Noun +Sg ↔ bu+Pron +A3sg +Pnon +Nom uç+Verb +Pos ˆDB+Adj+PresPart uçak+Noun +A3sg +Pnon +Nom

D.3 Testing Subset

1. a+Det +Indef +Sg pig+Noun +Sg come+Verb +Pres +3sg ↔ bir+Num+Card domuz+Noun +A3sg +Pnon +Nom gel+Verb +Pos +Aor +A3sg
2. all+Det +Pl school+Noun +Pl ↔ bütün+Adj okul+Noun +A3pl +Pnon +Nom
3. four+Num+Card car+Noun +Pl ↔ dört+Num+Card araba+Noun +A3sg +Pnon +Nom
4. all+Det +Pl pig+Noun +Pl ↔ bütün+Adj domuz+Noun +A3pl +Pnon +Nom
5. a+Det +Indef +Sg black+Adj car+Noun +Sg ↔ bir+Num+Card siyah+Adj araba+Noun +A3sg +Pnon +Nom
6. at+Prep least+Adv two+Num+Card car+Noun +Pl ↔ en+Adverb +AdjMdfy az+Adverb +AdjMdfy iki+Num+Card araba+Noun +A3sg +Pnon +Nom
7. at+Prep most+Adv two+Num+Card car+Noun +Pl ↔ en+Adverb +AdjMdfy çok+Adverb +AdjMdfy iki+Num+Card araba+Noun +A3sg +Pnon +Nom
8. at+Prep least+Adv one+Num+Card car+Noun +Sg ↔ en+Adverb +AdjMdfy az+Adverb +AdjMdfy bir+Num+Card araba+Noun +A3sg +Pnon +Nom
9. at+Prep most+Adv one+Num+Card car+Noun +Sg ↔ en+Adverb +AdjMdfy çok+Adverb +AdjMdfy bir+Num+Card araba+Noun +A3sg +Pnon +Nom

10. at+Prep least+Adv three+Num+Card notebook+Noun +Pl ↔ en+Adverb +AdjMdfy az+Adverb +AdjMdfy üç+Num+Card defter+Noun +A3sg +Pnon +Nom
11. at+Prep most+Adv three+Num+Card notebook+Noun +Pl ↔ en+Adverb +AdjMdfy çok+Adverb +AdjMdfy üç+Num+Card defter+Noun +A3sg +Pnon +Nom
12. ali+Noun +Prop +Sg +Part +Gen book+Noun +Sg ↔ ali+Noun +Prop +A3sg +Pnon +Gen kitap+Noun +A3sg +P3sg +Nom
13. every+Det +Sg pig+Noun +Sg ↔ her+Adj domuz+Noun +A3sg +Pnon +Nom
14. every+Det +Sg car+Noun +Sg ↔ her+Adj araba+Noun +A3sg +Pnon +Nom
15. four+Num+Card brown+Adj car+Noun +Pl ↔ dört+Num+Card kahverengi+Adj araba+Noun +A3sg +Pnon +Nom
16. three+Num+Card apple+Noun +Pl ↔ üç+Num+Card elma+Noun +A3sg +Pnon +Nom
17. cop+Noun +Pl be+Verb +Pres +Pl approach+Verb +Prog ↔ polis+Noun +A3pl +Pnon +Nom yaklaş+Verb +Pos +Prog1 +A3pl
18. cop+Noun +Pl be+Verb +Pres +Pl not+Adv approach+Verb +Prog ↔ polis+Noun +A3pl +Pnon +Nom yaklaş+Verb +Neg +Prog1 +A3pl
19. thief+Noun +Pl be+Verb +Pres +Pl steal+Verb +Prog ↔ hırsız+Noun +A3pl +Pnon +Nom çal+Verb +Pos +Prog1 +A3pl
20. thief+Noun +Pl be+Verb +Pres +Pl not+Adv steal+Verb +Prog ↔ hırsız+Noun +A3pl +Pnon +Nom çal+Verb +Neg +Prog1 +A3pl
21. three+Num+Card cop+Noun +Pl ↔ üç+Num+Card polis+Noun +A3sg +Pnon +Nom
22. girl+Noun +Pl be+Verb +Pres +Pl approach+Verb +Prog ↔ kız+Noun +A3pl +Pnon +Nom yaklaş+Verb +Pos +Prog1 +A3pl
23. cop+Noun +Pl will+Aux not+Adv approach+Verb +Pres +Non3sg ↔ polis+Noun +A3pl +Pnon +Nom yaklaş+Verb +Neg +Fut +A3pl
24. thief+Noun +Pl will+Aux not+Adv steal+Verb +Pres +Non3sg ↔ hırsız+Noun +A3pl +Pnon +Nom çal+Verb +Neg +Fut +A3pl
25. cop+Noun +Pl will+Aux approach+Verb +Pres +Non3sg ↔ polis+Noun +A3pl +Pnon +Nom yaklaş+Verb +Pos +Fut +A3pl
26. thief+Noun +Pl will+Aux steal+Verb +Pres +Non3sg ↔ hırsız+Noun +A3pl +Pnon +Nom çal+Verb +Pos +Fut +A3pl
27. girl+Noun +Pl will+Aux approach+Verb +Pres +Non3sg ↔ kız+Noun +A3pl +Pnon +Nom yaklaş+Verb +Pos +Fut +A3pl
28. cop+Noun +Pl will+Aux go+Verb +Pres +Non3sg tomorrow+Adv ↔ polis+Noun +A3pl +Pnon +Nom yarın+Adverb git+Verb +Pos +Fut +A3pl
29. these+Det +Pl cop+Noun +Pl ↔ bu+Adj polis+Noun +A3pl +Pnon +Nom
30. those+Det +Pl cop+Noun +Pl ↔ şu+Adj polis+Noun +A3pl +Pnon +Nom
31. two+Num+Card cold+Adj cop+Noun +Pl ↔ iki+Num+Card soğuk+Adj polis+Noun +A3sg +Pnon +Nom
32. four+Num+Card cold+Adj beer+Noun +Pl ↔ dört+Num+Card soğuk+Adj bira+Noun +A3sg +Pnon +Nom
33. three+Num+Card cold+Adj whiskey+Noun +Pl ↔ üç+Num+Card soğuk+Adj viski+Noun +A3sg +Pnon +Nom
34. i+Pron+Pers +Nom +1sg be+Verb +Pres +1sg a+Det +Indef +Sg thief+Noun +Sg ↔ ben+Pron +A1sg +Pnon +Nom bir+Num+Card hırsız+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A1sg
35. i+Pron+Pers +Nom +1sg be+Verb +Pres +1sg a+Det +Indef +Sg boy+Noun +Sg ↔ ben+Pron +A1sg +Pnon +Nom bir+Num+Card oğlan+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A1sg

36. you+Pron+Pers +Nom +2sg be+Verb +Pres +Pl a+Det +Indef +Sg boy+Noun +Sg ↔ sen+Pron +A2sg +Pnon +Nom bir+Num+Card oğlan+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A2sg
37. you+Pron+Pers +Nom +2sg be+Verb +Pres +Pl a+Det +Indef +Sg thief+Noun +Sg ↔ sen+Pron +A2sg +Pnon +Nom bir+Num+Card hırsız+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A2sg
38. i+Pron+Pers +Nom +1sg be+Verb +Pres +1sg not+Adv a+Det +Indef +Sg thief+Noun +Sg ↔ ben+Pron +A1sg +Pnon +Nom bir+Num+Card hırsız+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A1sg
39. i+Pron+Pers +Nom +1sg be+Verb +Pres +1sg not+Adv a+Det +Indef +Sg boy+Noun +Sg ↔ ben+Pron +A1sg +Pnon +Nom bir+Num+Card oğlan+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A1sg
40. you+Pron+Pers +Nom +2sg be+Verb +Pres +Pl not+Adv a+Det +Indef +Sg boy+Noun +Sg ↔ sen+Pron +A2sg +Pnon +Nom bir+Num+Card oğlan+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A2sg
41. you+Pron+Pers +Nom +2sg be+Verb +Pres +Pl not+Adv a+Det +Indef +Sg thief+Noun +Sg ↔ sen+Pron +A2sg +Pnon +Nom bir+Num+Card hırsız+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A2sg
42. it+Pron+Pers +Nom +3sg be+Verb +Pres +3sg a+Det +Indef +Sg notebook+Noun +Sg ↔ o+Pron +A3sg +Pnon +Nom bir+Num+Card defter+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +Cop +A3sg
43. it+Pron+Pers +Nom +3sg be+Verb +Pres +3sg a+Det +Indef +Sg car+Noun +Sg ↔ o+Pron +A3sg +Pnon +Nom bir+Num+Card araba+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +Cop +A3sg
44. it+Pron+Pers +Nom +3sg be+Verb +Pres +3sg not+Adv a+Det +Indef +Sg notebook+Noun +Sg ↔ o+Pron +A3sg +Pnon +Nom bir+Num+Card defter+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +Cop +A3sg
45. it+Pron+Pers +Nom +3sg be+Verb +Pres +3sg not+Adv a+Det +Indef +Sg car+Noun +Sg ↔ o+Pron +A3sg +Pnon +Nom bir+Num+Card araba+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +Cop +A3sg
46. it+Pron+Pers +Nom +3sg be+Verb +Pres +3sg not+Adv a+Det +Indef +Sg pig+Noun +Sg ↔ o+Pron +A3sg +Pnon +Nom bir+Num+Card domuz+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +Cop +A3sg
47. it+Pron+Pers +Nom +3sg be+Verb +Pres +3sg a+Det +Indef +Sg school+Noun +Sg ↔ o+Pron +A3sg +Pnon +Nom bir+Num+Card okul+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +Cop +A3sg
48. ali+Noun +Prop +Sg +Part +Gen car+Noun +Sg ↔ ali+Noun +Prop +A3sg +Pnon +Gen araba+Noun +A3sg +P3sg +Nom
49. the+Det +Def +SP cold+Adj beer+Noun +Sg ↔ soğuk+Adj bira+Noun +A3sg +Pnon +Nom
50. the+Det +Def +SP cold+Adj beer+Noun +Pl ↔ soğuk+Adj bira+Noun +A3pl +Pnon +Nom
51. the+Det +Def +SP whole+Adj week+Noun +Sg ↔ bütün+Adj hafta+Noun +A3sg +Pnon +Nom
52. they+Pron+Pers +Nom +3pl be+Verb +Pres +Pl thief+Noun +Pl ↔ o+Pron +A3pl +Pnon +Nom hırsız+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A3pl
53. they+Pron+Pers +Nom +3pl be+Verb +Pres +Pl not+Adv thief+Noun +Pl ↔ o+Pron +A3pl +Pnon +Nom hırsız+Noun +A3sg +Pnon +Nom değil+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +A3pl

54. which+Pron+Wh +NomObl +3SP thief+Noun +Pl ↔ hangi+Adj +Ques hırsız+Noun +A3pl +Pnon +Nom
55. which+Pron+Wh +NomObl +3SP cop+Noun +Sg ↔ hangi+Adj +Ques polis+Noun +A3sg +Pnon +Nom
56. which+Pron+Wh +NomObl +3SP tailor+Noun +Pl ↔ hangi+Adj +Ques terzi+Noun +A3pl +Pnon +Nom
57. a+Det +Indef +Sg very+Adv sick+Adj girl+Noun +Sg ↔ çok+Adverb +AdjMdfy hasta+Adj bir+Num+Card kız+Noun +A3sg +Pnon +Nom
58. a+Det +Indef +Sg literate+Adj woman+Noun +Sg ↔ okur yazar+Adj bir+Num+Card kadın+Noun +A3sg +Pnon +Nom
59. red+Adj hair+Noun +Sg ˆDB+Adj+Ed woman+Noun +Sg be+Verb +Pres +3sg come+Verb +Prog ↔ kızıl+Adj saç+Noun +A3sg +Pnon +Nom ˆDB+Adj+With kadın+Noun +A3sg +Pnon +Nom gel+Verb +Pos +Prog1 +A3sg
60. personnel+Noun +Sg of+Prep the+Det +Def +SP year+Noun +Sg ↔ yıl+Noun +A3sg +Pnon +Gen eleman+Noun +A3sg +P3sg +Nom
61. tom+Noun +Prop +Sg +Part +Gen voice+Noun +Sg ↔ tom+Noun +Prop +A3sg +Pnon +Gen ses+Noun +A3sg +P3sg +Nom
62. tom+Noun +Prop +Sg +Part +Gen height+Noun +Sg ↔ tom+Noun +Prop +A3sg +Pnon +Gen boy+Noun +A3sg +P3sg +Nom
63. tom+Noun +Prop +Sg +Part +Gen size+Noun +Sg ↔ tom+Noun +Prop +A3sg +Pnon +Gen beden+Noun +A3sg +P3sg +Nom
64. the+Det +Def +SP baby+Noun +Sg be+Verb +PastSimp +Sg cry+Verb +Prog ↔ bebek+Noun +A3sg +Pnon +Nom ağla+Verb +Pos +Prog1 +Past +A3sg
65. the+Det +Def +SP plane+Noun +Sg be+Verb +PastSimp +Sg fly+Verb +Prog ↔ uçak+Noun +A3sg +Pnon +Nom uç+Verb +Pos +Prog1 +Past +A3sg
66. tom+Noun +Prop +Sg +Part +Gen watch+Noun +Sg ↔ tom+Noun +Prop +A3sg +Pnon +Gen saat+Noun +A3sg +P3sg +Nom
67. a+Det +Indef +Sg cup+Noun +Sg of+Prep coffee+Noun +Sg ↔ bir+Num+Card fincan+Noun +A3sg +Pnon +Nom kahve+Noun +A3sg +Pnon +Nom
68. a+Det +Indef +Sg empty+Adj chair+Noun +Sg ↔ boş+Adj bir+Num+Card sandalye+Noun +A3sg +Pnon +Nom
69. this+Det +Sg rise+Verb +Prog ˆDB+Adj+Zero sun+Noun +Sg ↔ bu+Pron +A3sg +Pnon +Nom doğ+Verb +Pos ˆDB+Adj+PresPart güneş+Noun +A3sg +Pnon +Nom
70. a+Det +Indef +Sg crash+Verb +Prog ˆDB+Adj+Zero plane+Noun +Sg ↔ bir+Num+Card düş+Verb +Pos ˆDB+Adj+PresPart uçak+Noun +A3sg +Pnon +Nom
71. this+Det +Sg crash+Verb +Prog ˆDB+Adj+Zero plane+Noun +Sg ↔ bu+Pron +A3sg +Pnon +Nom düş+Verb +Pos ˆDB+Adj+PresPart uçak+Noun +A3sg +Pnon +Nom
72. this+Det +Sg cry+Verb +Prog ˆDB+Adj+Zero baby+Noun +Sg ↔ bu+Pron +A3sg +Pnon +Nom ağla+Verb +Pos ˆDB+Adj+PresPart bebek+Noun +A3sg +Pnon +Nom
73. the+Det +Def +SP music+Noun +Sg be+Verb +PastSimp +Sg play+Verb +Prog ↔ müzik+Noun +A3sg +Pnon +Nom çal+Verb +Pos +Prog1 +Past +A3sg
74. the+Det +Def +SP scorpion+Noun +Sg be+Verb +PastSimp +Sg sting+Verb +Prog ↔ akrep+Noun +A3sg +Pnon +Nom sok+Verb +Pos +Prog1 +Past +A3sg
75. thief+Noun +Pl will+Aux steal+Verb +Pres +Non3sg letter+Noun +Pl ↔ hırsız+Noun +A3pl +Pnon +Nom mektup+Noun +A3pl +Pnon +Nom çal+Verb +Pos +Fut +A3pl
76. priest+Noun +Pl will+Aux ring+Verb +Pres +Non3sg bell+Noun +Pl ↔ rahip+Noun +A3pl +Pnon +Nom çan+Noun +A3pl +Pnon +Nom çal+Verb +Pos +Fut +A3pl
77. the+Det +Def +SP phone+Noun +Sg be+Verb +PastSimp +Sg ring+Verb +Prog ↔ telefon+Noun +A3sg +Pnon +Nom çal+Verb +Pos +Prog1 +Past +A3sg

78. the+Det +Def +SP nature+Noun +Sg be+Verb +PastSimp +Sg call+Verb +Prog ↔ doğa+Noun +A3sg +Pnon +Nom çağır+Verb +Pos +Prog1 +Past +A3sg
79. a+Det +Indef +Sg ring+Verb +Prog ˆDB+Adj+Zero bell+Noun +Sg ↔ bir+Num+Card çal+Verb +Pos ˆDB+Adj+PresPart çan+Noun +A3sg +Pnon +Nom
80. the+Det +Def +SP strange+Adj habit+Noun +Pl ↔ garip+Noun +A3sg +Pnon +Nom adet+Noun +A3pl +Pnon +Nom
81. it+Pron+Pers +Nom +3sg be+Verb +Pres +3sg a+Det +Indef +Sg watch+Noun +Sg ↔ o+Pron +A3sg +Pnon +Nom bir+Num+Card saat+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +Cop +A3sg
82. a+Det +Indef +Sg hard+Adj problem+Noun +Sg ↔ zor+Adj bir+Num+Card sorun+Noun +A3sg +Pnon +Nom
83. a+Det +Indef +Sg hard+Adj rock+Noun +Sg ↔ sert+Adj bir+Num+Card kaya+Noun +A3sg +Pnon +Nom
84. tom+Noun +Prop +Sg +Part +Gen mouth+Noun +Sg ↔ tom+Noun +Prop +A3sg +Pnon +Gen ağız+Noun +A3sg +P3sg +Nom
85. priest+Noun +Pl will+Aux buy+Verb +Pres +Non3sg food+Noun +Pl ↔ rahip+Noun +A3pl +Pnon +Nom yiyecek+Noun +A3pl +Pnon +Nom al+Verb +Pos +Fut +A3pl
86. john+Noun +Prop +Sg +Part +Gen mother+Noun +Sg ↔ john+Noun +Prop +A3sg +Pnon +Gen anne+Noun +A3sg +P3sg +Nom
87. these+Det +Pl interval+Noun +Pl ↔ bu+Adj aralık+Noun +A3pl +Pnon +Nom
88. this+Det +Sg department+Noun +Sg ↔ bu+Adj bölüm+Noun +A3sg +Pnon +Nom
89. the+Det +Def +SP girl+Noun +Sg be+Verb +PastSimp +Sg call+Verb +Prog ↔ kız+Noun +A3sg +Pnon +Nom çağır+Verb +Pos +Prog1 +Past +A3sg
90. boy+Noun +Pl will+Aux write+Verb +Pres +Non3sg paper+Noun +Pl ↔ oğlan+Noun +A3pl +Pnon +Nom makale+Noun +A3pl +Pnon +Nom yaz+Verb +Pos +Fut +A3pl
91. these+Det +Pl paper+Noun +Pl ↔ bu+Adj makale+Noun +A3pl +Pnon +Nom
92. this+Det +Sg paper+Noun +Sg ↔ bu+Adj makale+Noun +A3sg +Pnon +Nom
93. these+Det +Pl mouth+Noun +Pl ↔ bu+Adj ağız+Noun +A3pl +Pnon +Nom
94. the+Det +Def +SP child+Noun +Sg be+Verb +PastSimp +Sg call+Verb +Prog ↔ çocuk+Noun +A3sg +Pnon +Nom çağır+Verb +Pos +Prog1 +Past +A3sg
95. man+Noun +Pl will+Aux read+Verb +Pres +Non3sg paper+Noun +Pl ↔ adam+Noun +A3pl +Pnon +Nom makale+Noun +A3pl +Pnon +Nom oku+Verb +Pos +Fut +A3pl
96. which+Pron+Wh +NomObl +3SP portion+Noun +Sg ↔ hangi+Adj +Ques bölüm+Noun +A3sg +Pnon +Nom
97. it+Pron+Pers +Nom +3sg be+Verb +Pres +3sg a+Det +Indef +Sg song+Noun +Sg ↔ o+Pron +A3sg +Pnon +Nom bir+Num+Card şarkı+Noun +A3sg +Pnon +Nom ˆDB+Verb+Zero +Pres +Cop +A3sg
98. yellow+Adj car+Noun +Sg ↔ sarı+Adj araba+Noun +A3sg +Pnon +Nom
99. ali+Noun +Prop +Sg +Part +Gen plane+Noun +Sg ↔ ali+Noun +Prop +A3sg +Pnon +Gen uçak+Noun +A3sg +P3sg +Nom
100. three+Num+Card car+Noun +Pl ↔ üç+Num+Card araba+Noun +A3sg +Pnon +Nom