EXAMPLE BASED MACHINE TRANSLATION SYSTEM BETWEEN KAZAKH AND TURKISH SUPPORTED BY STATISTICAL LANGUAGE MODEL

KAZAKÇA VE TÜRKÇE DİLLERİ ARASINDA ÖRNEK TABANLI VE İSTATİSTİK MODEL DESTEKLİ MAKİNE ÇEVİRİ SİSTEMİ

Gulshat KESSİKBAYEVA

Prof.Dr. İlyas ÇİÇEKLİ
Supervisor

Submitted to Graduate School in Science and Engineering of Hacettepe University as a Partial Fulfillment to the Requirements for the Award of the Degree of Doctor of Philosophy in Computer Engineering

2016

This work named "**EXAMPLE BASED MACHINE TRANSLATION SYSTEM BETWEEN KAZAKH AND TURKISH SUPPORTED BY STATISTICAL LANGUAGE MODEL**" by **GULSHAT KESSIKBAYEVA** has been approved as a thesis for the degree of **DOCTOR OF PHILOSOPHY IN COMPUTER ENGINEERING** by the below mentioned Examining Committee Members.

Prof. Dr. Ferda NUR ALPASLAN
Head .....

Prof. Dr. İlyas ÇİÇEKLİ
Supervisor .....

Prof. Dr. Ahmet COŞAR
Member .....

Asst. Prof. Dr. Gönenç ERCAN
Member .....

Asst. Prof. Dr. Ayça TARHAN
Member .....

This thesis has been approved as a thesis for the Degree of **DOCTOR OF PHILOSOPHY IN COMPUTER ENGINEERING** by Board of Directors of the Institute for Graduate School in Science and Engineering.

Prof. Dr. Salih Bülent ALTEN
Director of the Institute of Graduate School of Science and Engineering

ETHICS

In this thesis study, prepared in accordance with the spelling rules of Institute of Graduate School in Science and Engineering of Hacettepe University,
I declare that

- all the information and documents have been obtained in the base of the academic rules
- all audio-visual and written information and results have been presented according to the rules of scientific ethics
- in case of using other Works, related studies have been cited in accordance with the scientific standards
- all cited studies have been fully referenced
- I did not do any distortion in the data set
- And any part of this thesis has not been presented as another thesis study at this or any other university.

24 May 2016
GULSHAT KESSIKBAYEVA

ABSTRACT

EXAMPLE BASED MACHINE TRANSLATION SYSTEM BETWEEN KAZAKH AND TURKISH SUPPORTED BY STATISTICAL LANGUAGE MODEL

Gulshat KESSIKBAYEVA

Doctor of Philosophy, Department of Computer Engineering

Supervisor: Prof.Dr. İlyas ÇİÇEKLİ

May 2016, 147 pages

Example Based Machine Translation System(EBMT) is a analogy-based type of Machine Translation(MT), where translation made according to aligned bilingual corpus. Moreover, there are a lot of different methodologies in MT and hybridization is also possible between these methods which focused on compounding the strongest sides of more than one MT approaches to provide better translation quality. There are two parts of Hybrid Machine Translation (HMT) such as guided part and information part.

Our work is guided by EBMT and a hybrid example based machine translation system between Kazakh and Turkish languages is presented here. Analyzing both languages at morphological level, then constructing morphological processors is one of the most important part of the system. Their morphological processors are used to obtain the lexical forms of the surface level words and the surface level forms of translation results at lexical level. Translation templates are kept at lexical level and they translate a given source language sentence at lexical level to a target language sentence at lexical level. Our bilingual corpora hold translation examples at surface level and their words are

morphologically analyzed by appropriate morphological analyzer before they are fed into the learning module.

Thus, translation templates are learned at morphological level from a bilingual parallel corpus between Turkish and Kazakh. Translations can be performed at both directions using these learned translation templates.

The system is supported by a statistical language model for the target language. Therefore, translation results are sorted according to both their confidence factors that are computed using the confidence factors of the translation templates used in those translations and statistical language model probabilities of those translation results. Thus, the statistical language model of the target language is used in the ordering of translation results in addition to translation template confidence factors in order obtain more precise translation results.

Our main aim with our hybrid example based machine translation system is to obtain more accurate translation results by pre-gained knowledge from target language resource. One of the reasons that we propose this hybrid approach is that monolingual language resources are more widely available than bilingual language resources. In this thesis, experiments show that we can rely on the combination of EBMT and SMT approaches, because it produces satisfying results.

Keywords: Example Based Machine Translation, Statistical Language Model, Machine Learning.

ÖZET

KAZAKÇA VE TÜRKÇE DİLLERİ ARASINDA ÖRNEK TABANLI VE İSTATİSTİK MODEL DESTEKLİ MAKİNE ÇEVİRİ SİSTEMİ

Gulshat KESSİKBAYEVA

Doktora, Bilgisayar Mühendisliği Bölümü

Danışman: Prof. Dr. İlyas ÇİÇEKLİ

Mayıs 2016, 147 Sayfa

Örnek Tabanlı Makine Çeviri Sistemi tercümenin paralel metinler kullanılarak elde edildiği benzerlik bazlı makine çeviri sistemi yöntemlerindendir. Makine çeviri sistemlerinin birden fazla çeşitleri mevcuttur ve tercümenin kalitesini arttırma adına metotlardan bir kaçının güçlü yönleri birleştirilerek hibrit yaklaşımların oluşturulması mümkündür. Genellikle hibrit yöntemlerin iki ana kısmından bahsedilir ki, bunlar güdümlü ve bilgi kısımlarıdır.

Bu çalışma Kazakça ve Türkçe dilleri arasında temelde Örnek Tabanlı Çeviri Sistemi sahip bir hibrit sistemdir. Her iki dilin biçim bilgisinin analizi ve ardından biçim bilgisi işlemcisinin oluşturulması sistemin önemli bileşenlerinden biridir. Biçim bilgisi işlemcisi kelimelerin yüzeysel biçiminden sözcük bazlı analiz biçimini elde etmek için ve aynı zamanda tercüme sonuçlarının sözcük bazlı analiz biçiminden yüzeysel kelimelerin elde edilmesinde kullanılır. Başka bir deyişle, tercüme kalıpları sözcük bazlı analiz seviyesinde tutulur ve işlenir. Yani, kaynak dilde sözcük bazlı analiz seviyesinde verilen cümle hedef dilde sözcük bazlı analiz seviyesinde sonuç üretir. Sistemde kullanılan paralel metinler yüzeysel kelime bazlı örnekler cinsinde tutulur ve her biri ilgili biçim bilgisi

işlemcisinden sonra öğrenme modülüne yüklenir. Sonuç olarak Kazakça ve Türkçe dilleri arasındaki tercüme kalıpları sözcük bazlı analiz seviyesinde öğrenilir. Öğrenilen kalıpları kullanarak sistem çift yönlü tercüme yapmaktadır.

Sistem tarafından üretilen sonuçlar örnek kalıpları açısından güvenirlilik faktör değerine göre sıralanmaktadır. Diğer yandan sistem kaynak dil açısından istatistik model desteklidir. Bu üretilen sonuçların sadece örnek kalıpları açısından değil de aynı zamanda istatistiksel cümle olasılığının hesaplanması ve böylece kaynak dil açısından geçerliliğin göz önünde bulundurmasıdır. Bu da sonuçların kaynak dilde geçerli ve daha düzgün tercüme sonucu anlamına gelmektedir.

Bu çalışmada hibrit örnek tabanlı çeviri sistemin kullanılmasındaki temel amaç kaynak dilde öncesinden hesaplanmış istatistik modelleri kullanarak daha geçerli ve düzgün sonuç elde etmektir. Bahsedilen hibrit sistem önerisinin nedenlerinden birisi tek dilde erişebilen kaynakların paralel işlenmiş metinlerin elde edilmesinden daha kolay ve olanaklı oluşudur. Deney sonucu olarak ta tatmin edici neticelere ulaşıldığından örnek tabanlı çeviri sistemlerin istatistiksel modellerle desteklenmesi istinad edilebilir bulunmaktadır.

Anahtar Kelimeler: Örnek Tabanlı Çeviri Sistemleri, İstatistiksel Modeller, Makine Öğrenimi.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my advisor Prof. Dr. İlyas ÇİÇEKLİ for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. I could not have imagined having a better advisor and mentor for my research in Ph.D study.

Besides my advisor, I would like to thank the rest of my thesis committee members, Prof. Dr. Ferda Nur ALPASLAN, Prof. Dr. Ahmet COŞAR, Prof. Dr. Hayri SEVER, Asst. Prof. Dr. Gönenç ERCAN, and Asst. Prof. Dr. Ayça TARHAN for reviewing this thesis and providing me with helpful feedbacks. I also would like to thank them for their valuable comments and suggestions.

Finally, I would like to acknowledge friends and family who supported me during my time here and throughout my life. First and foremost, my sincere thanks go to my family members, for their endless love and support, especially my parents who have always done their best for me. I could not have achieved this without their unlimited help and encouragements. I am indebted to my husband whose love, encouragement, unfailing support and patience made all this work possible. Also, my lovely son *Amir Ali* and daughter *Alima Nur* have always been a great source of passion and motivation. I am lucky to have met Öznur Türkmen, and I thank her for her friendship, love, and unyielding support.

24 May 2016

Gulshat KESSIKBAYEVA

CONTENTS

Pages

FIGURES

Pages

TABLES

Pages

SYMBOLS AND ABBREBIATIONS

Symbols

Abbreviations

NLP	Natural Language Processing
MT	Machine Translation
EBMT	Example Based Machine Translation
SMT	Statistical Machine Translation
RBMT	Rule Based Machine Translation
HMT	Hybrid Machine Translation
TM	Translation Model
LM	Language Model
L1	Source Language
L2	Target Language
CF	Confidence Factor
MRR	Mean Reciprocal Rank
BLEU	Bilingual Evaluation Understudy
LC	Left Context
RC	Right Context
FST	Finite State Transducer
SOV	Subject-Object-Verb
Sg	Singular
Pl	Plural

Nom	Nominative
Loc	Locative
Dat	Dative
Gen	Genitive
Acc	Accusative
Abl	Ablative
Ins	Instrumental
A3sg	Agreement of Third Person Singular
A3pl	Agreement of Third Person Plural
Pnon	None of Possesive Affix
P3Sg	Possessive Third Person Singular
Adj	Adjective
Adv	Adverb
Conj	Conjugate
Prop	Proposition
WTBL	Most Likely Tag of Word Table
STBL	Most Likely Tag of Suffix Table
TTL	Translation Template Learning
DB	Derivational Boundary

1. INTRODUCTION

1.1. MACHINE TRANSLATION

Translation between natural languages has a vitality for humankind from the strach and its history is as old as encounters of people from different communities with different languages such as travelers, traders, artisans, politicians, or missionaries who wished to communicate, to explain their messages, or to reach an an agreement with the foreigners. Nowadays, Machine Translation is gaining more importance as world is moving towards globalization and cultural socialization.

In fact, Machine Translation (MT) concept emerged when scientist such as Warren Waver [1] thought about the idea of automatic translation of texts between natural languages shortly after the end of the World War II. By this period, translation was treated as code-breaking system which uses a natural language as a code. Therefore, machine translation was seen as a matter of finding some systematic translation approach inspired by the cryptanalysis techniques developed at that time. Since the complexities of natural language arosed from speakers' mental influence with cultural, historical, geographical effects, it can not be replaced by codes and translated as code-breaking system. Therefore, MT result is still out of reach of translation made by expert human translators. However, MT is successful on restricted domains such as translating a weather report or legal documents which are highly standardized texts, or a text with no well-defined grammar properness are required. There are significant research efforts on this field with increased number of developing technologies. Thus, in near future MT is expected to approach the level of anticipated value.

There are two main categorization of MT is defined according to different aspects. The scope of the first categorization is the architectural basis on which the MT systems are built upon whereas the second categorization is defined according to the obtained data for translation process. In the first categorization,

MT systems vary in the level to which they analyze their inputs and it is shown in Figure 1, as a pyramid diagram [2].

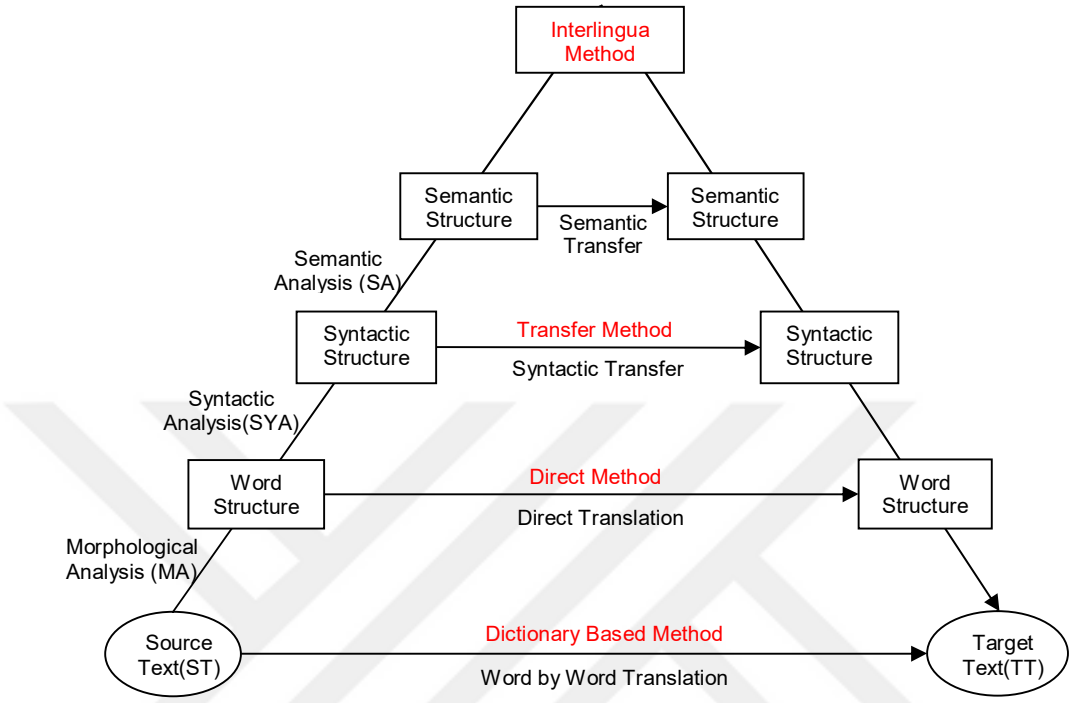


Figure 1. Vauquois' Pyramid for Machine Translation

As it can be observed in Vauquois' Pyramid given in Figure 1, the objective of MT systems is to convert souce language structures to target language structures. MT systems can perform transformations from source language to target language at different levels. When we move to upper levels of the pyramid, we have to perform more complicated analysis on source language texts and genaration methods on target language structures. MT methods at lower levels are simpler than methods at upper levels.

In lowest level which is called Dictionary-Based Method is the simplest way of translating a phrase by replacing its translation in given bilingual dictionary. For that reason, usually this method is not included to the existent list of methods, however it is translation also and to some extent it is helpful in translation of phrases but not sentences.

Then, Direct translation is the next level where only dictionary and a few simple word-ordering rules are used. Here translation is performed by replacing each

word in source text by its most common usage in target text. It is the simplest way of translation with minimal analysis of input text and thus, has limited success rate.

Transfer-based translation category is the next category to upward direction which includes morphological analysis and syntactic analysis. In this approach, the source text is first converted into an internal representation, which is then transformed into a corresponding internal representation with relation of associated source. Transfer rules are often linguistic based rules.

On the top edge of the pyramid, interlingua based translation approach which is also called as Knowledge Based Machine Translation (KBMT) system, is illustrated, where a source text is morphologically, syntactically and semantically analyzed and finally converted into an interlingua representation which is independent both from the source and the target languages. The disadvantage of an interlingua approach is its expensive cost, due to a semantical analysis which requires real-word knowledge with its representation, efficient acquisition and storage expenses.

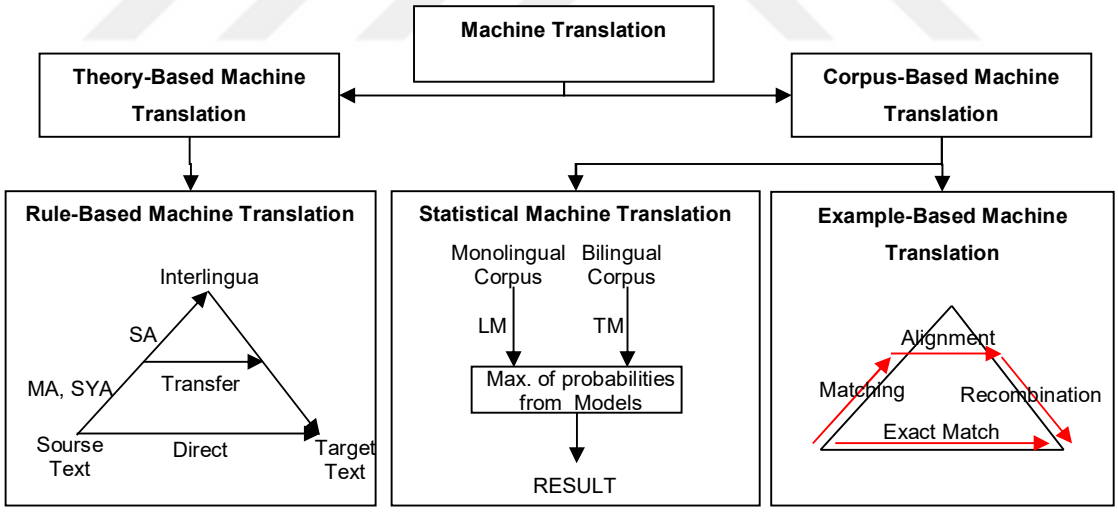


Figure 2. Classification of Machine Translation

The second categorization of MT depends to acquisition of knowledge inorder to perform translations. In other words, in translation process, there are mainly two basic stages such as interpretation of text in source language and generating an equivalent text in target language using automatic techniques [3]. These stages are achieved by predefined knowledge from source and target

languages. Somers classified different machine translation types according to their possession of knowledge, and these groups are theory or rule-based machine translation systems and corpus based machine translation systems [4] as illustrated in Figure 2.

1.1.1. Rule Based Machine Translation

RBMT is based on linguistic rules and in translation process, first input text is analyzed morphologically, syntactically and semantically [3] [5] [6]. Then output text is generated via structural conversions based on internal structure. These mentioned procedures are using dictionary and grammar which is defined and obtained by linguists. There are several approaches in RBMT as defined in Figure 2, such as Direct, Transfer Based and Interlingua approaches.

1.1.1.1. Direct Approach

As mentioned before, in this stage a text of source language are translated according to dictionary and some low-level rules without additional intermediate representation. Anusaarka is a RBMT system based on direct approach [7].

1.1.1.2. Transfer Based Approach

In transfer-based approach, a source language text is represented by intermediate representations with grammar rules and dictionaries. Then, equivalent representations are generated according to target language. Thus, it is less language specific approach of RBMT system than the Direct approach and it has three major modules such as analysis, transfer and synthesis.

First, *analysis* of source language text is done according to linguistic information as morphologic, part of speech, syntax and semantic analyses. Syntactic structures are derived from a source language for closely related languages in same language family. Semantic structures are derived for non-related languages from different families.

Then, in *transfer* module previously derived syntactic and semantic structures are transferred with transfer rules into syntactic and semantic structure of target language.

At last, *synthesis* module is responsible for replacing constituents of a source language with those in target language. Thus, there is a dependency between

involved language pairs. However, in Eurotra project [8] two independent monolingual dictionaries were suggested.

1.1.1.3. Interlingua Approach

In this approach the dependency to language pairs are minimized and text is converted to intermediate language which are not bounded to any language involved in the translation. Thus, only analysis and synthesis are needed for this approach. This module has an application such as UNITRAN[9] system which uses both formalization of syntactic and lexical distinctions .

1.1.1.4. Some Applications of RBMT

In fact, until the end of the 1980s the MT research was mainly based on linguistic rules such as: syntactic analysis, lexical, lexical transfer, syntactic generation and morphology rules. An information of projects and systems which use RBMT in last decades are given in [6], and briefly summarized here as follows. Some significant transfer systems of the 1980s such as Eurotra, Ariane, , Mu, Metal, and SUSY use mostly rule-based approach. Also there were some interlingua systems such as DLT and Rosetta which has a rule based approach as a basis of their system. There are some rule-based projects which are continued into 1990s such as the Catalyst project at Carnegie-Mellon University, CAT2 system at Saarbrücken, a project based on the linguistic theory at the University of Maryland, and Pangloss project which is an ARPA-funded research [3] [6]. Also, Nagao first started to work on rule based machine translation system, then he proposed another approach,which in advance leaded to example based machine translation [10].

1.1.2. Statistical Machine Translation

The emergency of new methods and strategies which are called ‘corpus-based’ methods brings the end of the ascendancy of the rule-based approach influence on MT. As mentioned before statistical machine translation systems are categorized as corpus based machine translation systems. First statistical machine translation idea was introduced by Warren Weaver in 1949, then it is re-introduced in the late 1980s by researchers of IBM Research Center in the Candide project reported in 1988 and developed in 1993 [11]. The Candide project uses statistical methods such as analysis and generation without any

linguistic rules. A typical RBMT system uses manually collected linguistic translation rules, whereas a SMT system infers translation models from bilingual corpus and monolingual corpus which possess information about target language in the form of statistical language models. Thus, there are two main components of SMT such as Translation Model (TM) which uses bilingual corpora to infer statistical model of translation and Language Model (LM) to reordering the output of translation according to target language and SMT system architecture is given in Figure 3 as defined in [12].

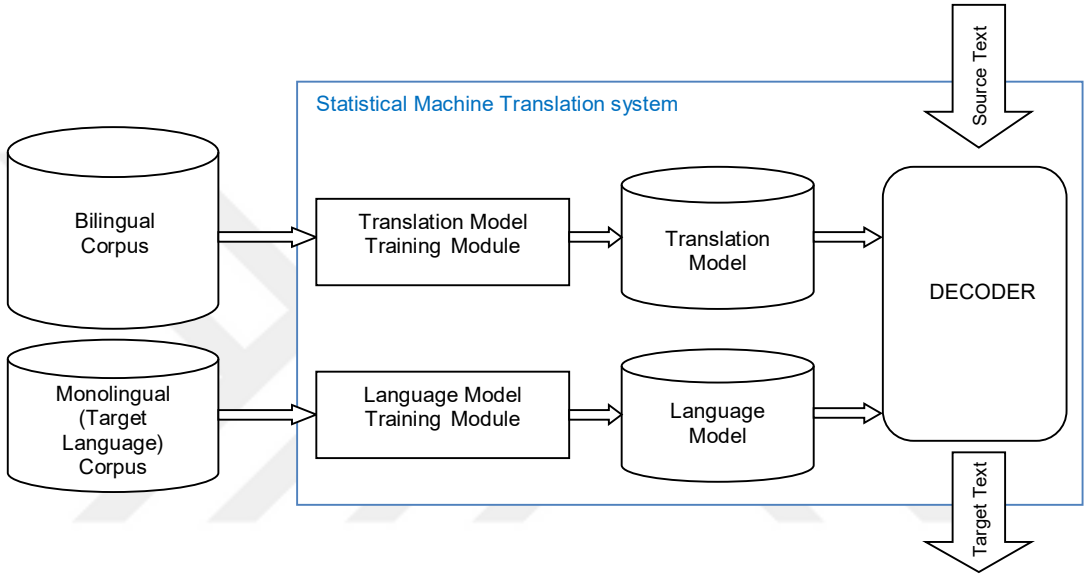


Figure 3 SMT Architecture

In other words, the TM is trained by bilingual data and stands for the possible word translations, which consist of sentence pairs in both languages involved in translation process. The LM provides the sentence accuracy in target language and is trained by the target language data. The decoder gives most likely outcome sentence in target language by searching a large amount of hypotheses using these two models.

1.1.2.1. Translation Model

A translation model that assigns a conditional probability P(S,T) to any source/target pair of sentences. Here $S=s_1 \ldots s_m$ is sentence in source language and $T=t_1 \ldots t_n$ is a sentence in target language of length m and n. The parameters of this model will be estimated from the translation examples from bilingual corpora. It is expected that this probability will be very small for the

translations that are not correct. Sentence as given S in source language and seeking for T in target language in order to maximize P(S, T).

There are mainly three kinds of translation model of SMT such as word-based, phrase-based and syntax-based translation models.

Word-Based model is reordering words which is a fundamental unit. Aligned words used by related algorithms to achieve most accuracy in translation process. However, natural language usually has more than one meaning for a word and its other features such as compound words, idioms constitute an obstackle for this type of model.

In Phrase-Based model the fundamental unit is a phrase or sequence of a sentence. Thus, a text in source language is segmented in phrase which is not necessarily linguistically valid and obtained phrases are then reordered. It's strengths in providing many to many translation, using local context in translation and allowing translation of non-compositional phrases. Some known phrase-based systems are word alignment induced phrase model [13], alignment templates [14] and joint phrase model [15] [16].

Syntax-Based model uses translation rules which define variables in source language, syntax binary tree in target language with variables and words represented as leaves and vector of likelyhood of pair language. This is the translation of syntactic features rather than single words or phrases. Lui and Gildea in their work have studied about semantic roles for improving the syntax-based approach [17].

1.1.2.2. Language Model

A language model that assigns a probability P(R) for any sentence $R = r_1 \ldots r_k$ in target language. The parameters of the language model can potentially be estimated from very large quantities of target language data. It is expected that this probability will be very small for the sentences that are not valid in target language. Thus the goal of language model is detecting a good translation result. In order to detect good translation result various methods have been discussed. These methods are n-grams which exist in several methodology such as bigrams, trigrams, 4-grams, 5-grams and 6-grams, etc. Successfull web n-grams as feature were discussed by Koehn and Knight [18] and an

exploiting non-parallel corpora to improve SMT is presented by Munteanu and Marcu [19]. In thesis, language model component of statistical machine translation is used to find more valid results in target language and benefited from bigrams method. See Chapter 8.

1.1.3. Example Based Machine Translation

Another major type of corpus-based approach is an example based machine translation. An EBMT system learns translation rules from bilingual corpus. History of the EBMT begins from the work of Makato Nagao presented in 1981 [10] and it served as a foundation for EBMT by the analogy principle where the linguistic knowledge is extracted from previous translation examples. So far researchers are trying to define and differentiate an EBMT system from other traditional ones. However, it is obvious that it can not be differentiated by strict lines, because there are multiple techniques which are reformed from other approaches such as methods used in RBMT and SMT. On the other hand, being analogy-based makes EBMT different from other translation systems [20]. For the significance of the topic if to quote the phrase of Nagao related with an issue given in [21]:

> ”Man does not translate a simple sentence by doing deep linguistic analysis, rather, man does translation, first, by properly decomposing an input sentence into certain fragmental phrases,then by translating these phrases into other language phrases, and finally by properly composing these fragmental translations into one long sentence. The translation of each fragmental phrase will be done by the analogy translation principle with proper examples as its reference.” (Nagao, 1984)

Makoto Nagao stated that EBMT is mainly suited to translation between two languages which are totally different from each other, such as English and Japanese. Thus, several well-structured sentences in target language can be obtained from translation of a sentence in source language, therefore, no deep linguistic analysis is required as for RBMT is needed. Indeed, EBMT is

applicable and more suitable than other MT systems when it is true for one of the following conditions, such as:

a) Formation of translation rule is difficult

b) Representation of special case which can not be described by general rule

c) Translation can not be performed in a compositional way from target words[22]

Also, it is known that EBMT is an expedient solution for the problem where rule-based approach was too difficult to preserve the structure [23] which is peculiar to MT systems. It is because many EBMT systems do not execute any structure. In fact, the range of languages for which EBMT is applicable is quite wide and many systems have been developed so far extensively. There are many variations among developed EBMT systems where some are least rule-based while some are most rule-based systems. Classification of developed systems can be divided into groups as given in [20], such as systems using tree derivations, run-time systems and template-driven systems. Derivation tree systems include developed approaches which are pre-compiled preparations of templates with more structure than in templates-driven systems. This systems defined in the work of Watanabe et al [26], Quirk and Menezes [27] and etc. Run-time approaches presented by the work of Eiichiro Sumita [22] where semantic distances are calculated by using thesauri and dynamic-programming matching. As for template-driven systems, the methods of extracting templates from bilingual example corpora are described by Cicekli and Guvenir in [24], where templates are used in the form of words with their POS tags for English and Turkish languages involved system. Also, Brown presented the induction of transfer rules using templates in [25]. All mentioned systems have in common basic stages of EBMT such as matching, aligning and recombination which distinguishes it from RBMT and SMT. These stages are to be discussed in scope of translation process of EBMT in Section 1.1.3.1.

1.1.3.1. Translation Process of EBMT

Nagao defines the translational process as if simple and short sentences in source language with their correspondences in target language is given to someone and he becomes able to translate new sentences by memorizing previously given pairs. There are three major statements of translation process in EBMT such as:

- Matching parts of translation pairs in database
- Identifying corresponding translation parts
- Bring together results from previous steps to obtain the target text

It is instructive to refer to familiar pyramid by Vauquois given in Figure 1 and adopted for EBMT in Figure 2. The text in source language is analyzed and replaced by the matching of input against the example set. In alignment stage, after the relevant examples are selected, the corresponding fragments from target text are choosen. In last stage which is recombination phase, appropriate fragments found in target text are combined to form a legal text in target language [20].

1.1.3.2. Matching Phase of EBMT

The most important step of translation is matching phase. In this step, the best match example is searched in database for a given source sentence. There are some common methods of matching such as character-based, word-based, annotated word-based, structure-based and parsing-based matching. In character-based method a distance or similarity measure is kept and matching depends only on that measure. A discriminatory work of Sato given in [28] by introducing a simple linguistic dimension to the matching process.

In word-based matching method, matching is said to be done when the words in the source string can be replaced by near synonyms in the translation pairs and classical similarity measure is proposed by Nagao [10] in its early stage. Also, considerable important illustration of this type of matching is issued by Sumita and Iida [22].

In annotated word-based matching part of speech tags are used. It means examples are kept partially parsed. This kind of matching makes use of annotated tree structure where explicit links are kept for correspondences. Usage of part of speech tags, really contributes a lot to the sentence composition phase. The measure that takes function words into account and makes use of POS tags are proposed by Cranias et al, in [29].

In structure-based matching type examples are assumed to be stored as structured objects and process involves a complex tree-matching affair which requires considerable computational cost. Development approaches are worked out by Maruyama and Watanabe in [30]. Also, there is parsing based matching type where example storage achieved by generalization. This method is used to match the best example to translate the given text. An alternative approach is described by Nirenburg et al [31] and Collins [32].

1.1.3.3. Adaptation and Recombination Phase of EBMT

Matching phase will select correct example pairs to use for translation while adaptation phase clarifies which parts of pairs are to be used and specifies the fragments of the examples that are to be used for translation. Assume that we have the sentence 1.10 to be translated into target language. From matching step we have correctly found two examples in 1.11 that suits our input string. After adaptation we will have the underlined fragments that are to be recombined to get the target sentence as explained in [33].

She reads a book on medicine (1.10)

<u>She reads</u> a newspaper

I borrow <u>a book on medicine</u> (1.11)

In order to ensure to recombine the right parts of examples some approaches store examples as tree structures and thus, corresspondences between fragments are explicitly defined. For instance, the recombination is a kind of tree unification in Sato's work [34] and Watanabe [35], [36] implements a flexible kind of graph unification called "gluing" [4].

For example, with a generalized example learned from given example set in 1.12. we are given sentence in 1.13, where X^1 is a correspondence of Y^1, thus they are translations of each other:

I will eat an apple↔elma yiyeceğim

I will eat a meal↔yemek yiyeceğim (1.12)

I will eat X^1 ↔ Y^1 yiyeceğim (1.13)

I will eat an orange (1.14)

Then, given sentence in 1.14 can be translated as in 1.15, if the correspondence given in 1.16 is known:

Portakal yiyeceğim (1.15)

orange↔portakal (1.16)

As we have the fragments that are to be combined to get the translated string, it seems pretty easy to get just by concatenation.

On the other hand, there is a boundary friction problem for some types of languages which have grammatical inflection to indicate syntactic function. It is especially big concern for agglutinative languages where every word carries its inflections and derivations to indicate grammatical cases. This problem is easied with POS tags as worked by Cicekli and Guvenir in [24]. Therefore, in this set of examples adaptation will be carried out for every tag of a word.

For the example above we get the correct result for English, because English is a little or no inflectional language. For other languages like Kazakh and Turkish which carry strong inflection property, yet remains some problems which can be alleviated by matching, aligning and recombining in POS tagged form. Detailed examples are given in Section 5.

Dealing each word with its POS tag is resulted in some kind of tackles to consider about such as storage, computational cost and an efficiency of algorithms additional to some fundamental problems of EBMT system. Moreover, having parallel aligned corpora and size, granularity and suitability of examples are fundamental problems of EBMT that have direct effect on

performance of EBMT process according to Somers [4] and will be discussed in detail in next Section.

1.1.3.4. Underlying problems of EBMT

Parallel Corpora

As mentioned above EBMT is corpus-based translation system, for this reason building an aligned corpora is the main thing that is needed. EBMT is considered to be best suited for a sublanguage approach. Nowadays, we have plenty multilanguage corpora which can be used by system. However, not all these system can properly meet the sublanguage criterion. After a suitable corpora is defined, there is a deal of aligning it. In order to easy the tackle example database can be built manually as it is in our case.

Size of Examples

While considering about size of an example database, it will worth to bother about the way of its storage and usage, because these are main factors to effect the size infact. In many reports, adding examples shows linear improvement in translation accuracy, but it is assumed that there is some limit after which adding examples will not enhance the quality of translation system [4].

Granularity of Examples

It is suggested from translation studies that even human translators work with smaller sentences. It is valid for EBMT also, that sentences should have easy determinable boundaries and simple structures often monoclausal property. According to Cranias [29] and Nirenburg [31], trade-off between legth and similarity of examples should be considered, because longer matched passages a valid threat for the probability of a complete match, while the shorter passages can cause a rise of an ambiguity.

Suitability of Examples

Suitability of examples is an important issue for EBMT system, because naturally occuring text or corpora will include considerably many overlapping examples that will load system with an extra baggage or force to make choice

by producing ambiguity. However, the system previously worked out by Oz and Cicekli [37] involves frequency sensitive similarity metric called as confidence factor which is employed by our system as well and presented in detail in Section 5.5.

1.1.4. Hybrid Machine Translation

It can be seen that MT has various approaches and it is a multidisciplinary field. The challenge is that translation can be solved by taking into account various points of view including linguistics and statistics. Hybrid methodologies are arised because of the existence of different approaches and focused on joining the strong sides of two or more MT approaches to provide better translation quality. There are two parts of HMT such as guided part and information part. If guided part is RBMT, then HMT is called Hybridization guided by RBMT and if guided part is corpus-based SMT or EBMT then it is called as Hybridization guided by corpus-based MT. In our case HMT is guided by corpus-based MT. It means main core of translation process is in EBMT, then RBMT with SMT are used as information part of the system. Moreover, core model of corpus-based MT approaches can be supported by rules which are integrated to it. Our work similarly with work of Carl et al.[38] in which the RBMT system dynamically integrated into an EBMT system [39] with its morphology and syntax knowledge. SMT is used as information for valid results at language model level. See Chapter 2.

1.2. Linguistic Knowledge

There are some linguistics terminologies used in the thesis and for more comprehension of the topic, their descriptions are to be given in this section. *Morphology* is the study of the word which built up from morphemes and their meanings with their functions within a word. A word is described as the minimal free form in a language which has a meaning and can stand alone. However, a morpheme is a minimal unit in a language which has smallest linguistic sound and semantic meaning within a word. Also, as opposed to a word a morpheme may or may not stand alone. For instance, in the word “toy” there is a single morpheme and in the word toys there are two morphemes: “*toy*” and “*–s*”. The morphemes can be consisted of two subtypes such as stems and affixes. The

meaning of a word is defined by the stem and it is the main morpheme, while affixes are attached to the stem in order to define additional meaning. However, the new formed meaning of a word after attaching an affix is still stem related. In the previous example, toy is the stem and –s is the affix which assigns plural property to the word.

After affixes being attached to the words it is inflected or derivated. Inflection and derivation of a word differs in remaining on main type of a stem or changing it. A word has its Part-Of-Speech (POS) tag within a sentence. Inflection does not cause changes in POS tag of a new word whereas derivational morpheme changes meaning of the root word and it may change its POS tag. In agglutinative languages infinitely many affixes can be appended to a word and each affix is a morpheme. *Morphotactics* defines the rule for orderings of the morphemes. For example, a word *globe* with suffix *-al* becomes global, then with an other suffix *–ize* it becomes *globalize* and at last with suffix –ation it ends with globalization. All these orderings are rules of morphotactics which inflects or derives a word to a meaningful one. Moreover, city in plural form will be cities, whereas boy in plural becomes boys. Here, rules of orthography step in by defining how morphemes will be added to particular ones. For example, most nouns that end in a vowel and -y , in plural inflection just will be added –s, such as:

toy → toys; ray → rays; key → keys.

Nouns that end in a consonant and a letter –y, in plural inflection becomes –ies, such as:

body→ bodies; country→ countries; try→tries

In NLP, constructing a word according to morphotactic and orthographic rules is the process of generating a word. *A morphological parse* of a word can contain derivational and inflectional suffixes. *Morphological parsing* is a process of decomposing a given word to the morphemes and defining them. It must be able to distinguish between orthographic rules and morphological rules. As opposed to generating process this is called an analyzing a word. *Morphological processors* usually generate and analyze a given word.

1.3. Related Researches

Language is a main tool for communication and socialization for people from different communities. Due to its interrelation between culture, history and geography, interpretation and then translation of a natural language is a complex issue. As stated before, in this work translation system between Kazakh and Turkish languages is presented. Both languages belong to same language family, which is Ural-Altaic group of languages. They are closely related and have similar features, common morphological and grammatical structures. Thus, most of the time a semantic analysis and word order consideration are not required for translation process.

There are many works performed on working out morphologies of natural language. These works can be classified as rule based, statistical or data driven methods and hybrid methods. Rule based morphological analyzers with FSTs for many languages including Finnish, Swedish, Russian, English, Swahili and Arabic have been developed [40]. Moreover, many studies and researches have been done upon on morphological analysis of Turkic Languages. The morphological analysis of Turkish is performed by a Turkish morphological processor developed earlier [41] which uses morphology rules defined by Oflazer [42]. Affix types and grammatical names in Kazakh morphological processor worked out in thesis are also defined similarly to Turkish morphological processor [42] [41]. There is a rule-based morphology analysis of Crimean Tatar developed for translation system which involves Turkish to Crimean Tatar in 2001 by Altintas and Cicekli [43]. Moreover, there is a morphological analyzer of Turkmen language worked out by Tantug [44]. In addition a rule-based morphological analysis of Uygur was developed by Orhun [45]. Also, a freely available Morphological Analyzer for Turkish is proposed by Cagri [46]. Especially for Kazakh language there is a considerable increase in NLP related research areas. Analysis of inflectional affix of Kazakh Language was studied within the work of Kazakh segmentation system [47]. A finite state approach for Kazakh nominals are presented by Kairakbay [48]. In this thesis, we present rules of alternations specific for each case, rather than generalized form. It can bring to over loaded size of rules for all grammar. Washington et al. developed Finite-state morphological transducer for three Kypchak languages

[49] including morphology for Kazakh language with limited stem size in lexicon. Also, Mahambetov et al. worked on Kazakh morphology with data-driven method by evaluating on the large data set with 97% accuracy while certain language-specific issues are not considered [50]. In our thesis, rule based morphological processor for Kazakh language differs from above works in that: First, it gives deep analysis of a language with inflectional and derivational morphemes. Also, it covers nearly all language specific issues. Finally, it does not require huge wordbased data sets of Kazakh language for morphological processor. The coverage of our morphological analyzer is substantial and its accuracy is 99%. It only does not cover some loan words, technical words and proper nouns.

Morphological disambiguators can be categorized as statistical, rule-based and hybrid systems. Statistical methods [51] [52] create probabilistic models from morphologically tagged texts and use these models to disambiguate words by selecting most probable morphological parses. There is a statistical morphological analysis for Turkish worked out using n-gram models for inflectional and final tags of words [53]. Rule-based morphological disambiguators [54] [55] [56] use hand-crafted rules to select correct parses of words or eliminate some of illegal parses of words. Disambiguation rules can be also learned from tagged texts using transformation-based learning approaches [57]. Hybrid systems [58] use both statistical 2 knowledge and disambiguation rules in disambiguation process. Turkish morphological disambiguator developed [59] by Kutlu and Cicekli uses both transformation based approach and rule-based approach. The morphological disambiguator for Kazakh language described in the thesis use transformation-based approach and it is a variation of Brill tagger [57].

There were many researches and works performed on translations between closely related languages based on some translation rules [60] [43] [45] [61]. There are also rule-based and statistical translation systems between Kazakh to Tatar languages worked out in [62]. However, the system presented in this research is based on a previous EBMT system [63] [24] [64] [37] and this previous EBMT system is between English-Turkish languages which are not closely related languages. Example Based translation paradigm are applied

before to unrelated languages such as Turkish to English, Japanese to English. In this work we intend to have successful result from applying example based translation paradigm to closely related languages.

1.4. Thesis Outline

Our system is a hybrid EBMT system which learns translation templates from examples with parallel aligned bilingual corpora between Turkish and Kazakh languages and translates given text to target language by using previously learned templates. In our hybrid translation system, statistical language models (SLM) of those languages are created from monolingual corpora, and the SLM of the target language is also used in the ordering of translation results in addition to confidence factors of learned translation templates from bilingual corpus. On the other words, we still use EBMT approach in translations but we use both EBMT and SMT approaches in the ordering of translation results. However, there is no language pair which has well-defined word to word relationships among each other. For that reason there will be more than one translation result for a given sentence. For a better result we developed hybrid system which gives more accurate translation results by combining an example based translation system with a statistical language model.

Each step of the system is explained in detailed form in next sections and organization of this thesis is as following. In next chapter, the system architecture is given for overall view and for better comprehension of next chapters. In Chapter 3, Turkish and Kazakh morphology is explained. Chapter 4 gives, the details of the morphological disambiguation systems that are used in our EBMT system. In Chapter 5, learning of translation of templates is explained together with type associated template learning and confidence factor calculation. In Chapter 6, the translation process is presented briefly. Statistical ranking of translation results is defined in Chapter 7. In Chapter 8, test and analysis are commented. In Chapter 9, conclusion and future works are given.

2. ARCHITECTURE OF THE HYBRID EBMT SYSTEM

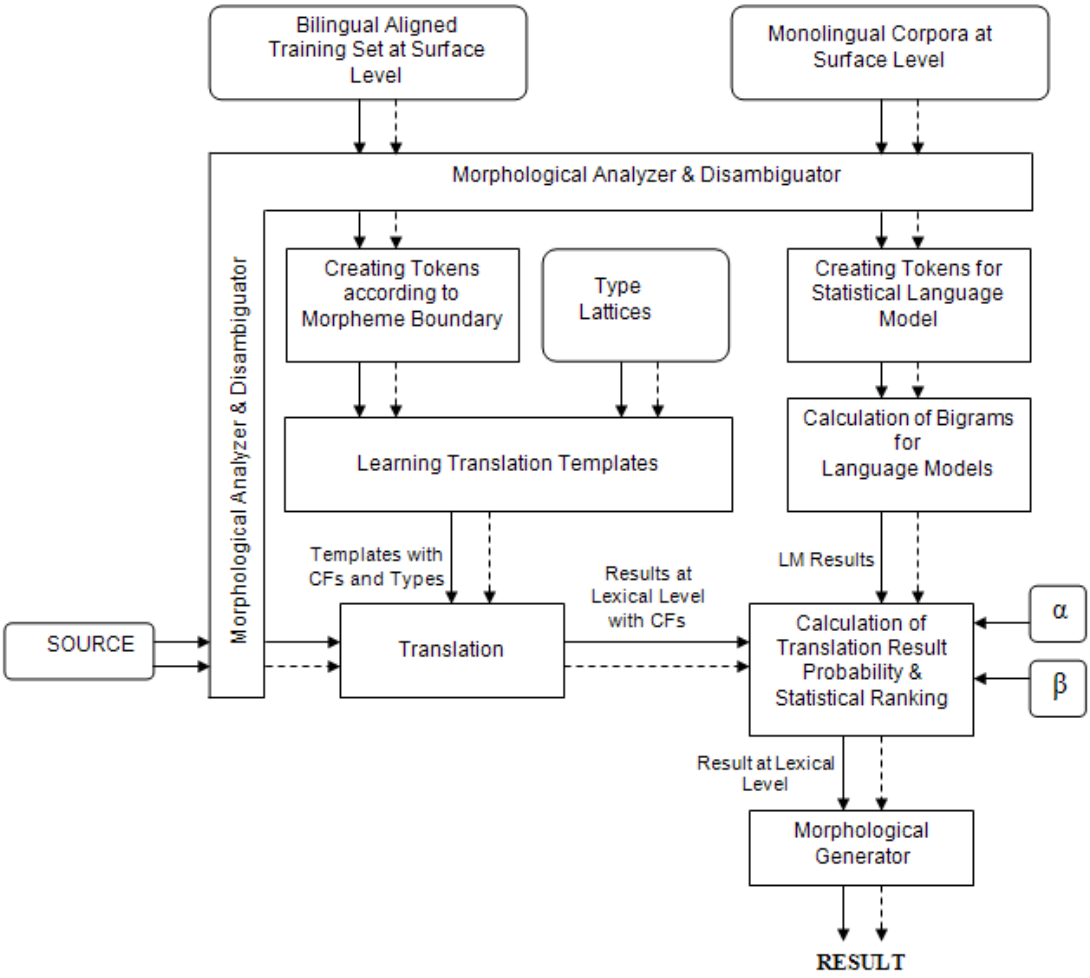


Figure 4. Architecture of EBMT system supported by SLM

Our translation system is a hybrid EBMT system which learns translation templates from translation examples with parallel aligned bilingual corpora between Turkish and Kazakh languages and translates a given source language text to target language using previously learned translation templates. The system described in this thesis builds upon the recent papers of Cicekli and Guvenir [60], [64]. In order to sort translation results, confidence factors of learned translation templates that are used. For this reason in addition to translation templates learned from bilingual corpora, SLMs of Turkish and Kazakh languages are inferred from their monolingual corpora. The general architecture of our hybrid EBMT system is given in Figure 1. In the system there are mainly 3 parts such as integrated model namely morphology and syntax

knowledge from the RBMT system which is morphological analyzer and disambiguator, information model which is language model of SMT system and core model which is EBMT system. Organization of this chapter is according to the order given above. First, general overview of integrated model which is morphological analyzer and disambiguator is given. After information model which is a language model of SMT system is briefly explained. Then, core model which is EBMT system is introduced. Thus, the general overview of the architecture is described in this chapter and the details of its certain parts are given in the rest of the thesis.

2.1. Morphological Analysis And Disambiguator

Turkish and Kazakh languages are both agglutinative languages and morphology plays an important role in the structures of their words and their sentences structures. Their morphological processors are used obtain the lexical forms of the surface level words and the surface level forms of translation results at lexical level. Translation templates are kept at lexical level and they translate a given source language sentence at lexical level to a target language sentence at lexical level. Our bilingual corpora hold translation examples at surface level and their words are morphologically analyzed by appropriate morphological analyzer before they are fed into the learning module in Figure 1. Similarly the source language sentence at surface level is analyzed by the source language morphological analyzer before it is sent into the translation module. Since SLMs are also created depending on the lexical features of words, surface level words in monolingual corpora are also morphologically parsed before SLMs are created. Turkish texts area analyzed using a morphological processor based on two level morphology description of Turkish [42]. Kazakh morphological processor [65] is developed using Foma environment [66] to be used in this translation system and explained in Chapter 3.

A word can have more than one morphological parse and the decreased ambiguity of text before translation process is important. On the other case, ambiguity of text is passed to the text step and it means having more unrelated translation result to be dealt after translation process. In order to disambiguate words morphologically, a Turkish morphological disambiguator [59] and a

Kazakh morphological disambiguator are used. Kazakh morphological disambiguator based on ideas on Turkish morphological disambiguator is developed to be used in this research and its some details are given in Chapter 4.

2.2. Language Model Of SMT

First of all, in order to accomplish translation supported with SLM there are background preparations such as defining parallel aligned bilingual corpora for learning part and collecting monolingual corpora for SLM part. In order create successful SLMs, it is important to have two large corpora that are analyzed and disambiguated in both languages. In our system the size of monolingual Turkish corpus is 300000 words and Kazakh corpus contains 100000 words. Then, these two large corpora are morphologically analyzed and disambiguated in order to prepare lexical level tokens for SLM part. In this portion, preparing a text by defining sentence boundaries, organizing different kinds of language models are performed. In order to create SLMs for different token types, bigrams of difference token types of sentence words are calculated using AntConc software [67]. After the creation of SLMs, the probability of a translation result can be calculated with a smoothing technique using a created SLM. The details of SLMs are described in Section 7.

2.3. EBMT As Core Model

Parallel bilingual text is morphologically analyzed and disambiguated with analyzers and disambiguators of related languages. Then, these results are prepared for learning process by creating tokens according to morphological boundaries. After that, the system is trained to learn translation templates from parallel aligned examples at lexical level. Also, morphological type lattices for both languages are given as input to the system in order to learn translation templates with morphologically typed variables in addition to translation templates with variables without type information. Here the morphological type lattice for Turkish morphemes was worked out earlier [63] and morphological type lattice for Kazakh morphemes is worked out similarly from the Kazakh morphological processor [65]. Then, the learning part infers translation templates which are rules for translation process as a result. Translation

templates are learned with confidence factors where a template confidence factor indicates the accuracy of that translation template for the training set. Using learned templates, bidirectional translations can be performed and translation results at lexical level together with their confidence factors are produced. As it can be seen in Figure 1, our hybrid EBMT system has mainly two parts, namely an EBMT part which is similar to an earlier developed system [63] [64] and a SLM part which is developed for this research. Translations can be done at both directions meaning that from Turkish to Kazakh language and from Kazakh language to Turkish.

A source language sentence is morphologically analyzed and disambiguated as mention before in order to obtain a disambiguated sentence at lexical level. Then the source language is translated to target language by the translation module using learned translation templates and more than one translation result can be produced. Although type associated rules [63] can produce better results since they can eliminated some of unsuited results using type information, still more than one result can be produced. In order to select more common or reliable translation results, confidence factors of those results are also calculated from confidence factors of templates used in translations [37]. Then, translation results can be sorted according to confidence factors by the EBMT part of the system. At this stage, a SLM of the target language is used to catch some better uses of equivalent text. In Figure 1, α and β weights indicate the effects of the EBMT part and the SLM part in the orderings of translation results. At the last step, the morphological generator of the target language converts result to the surface level. All these steps are shown in detailed form in Figure 1, and the usage of two arrow kinds in Figure 1 is for emphasizing bidirectional property of the system.

3. TURKISH AND KAZAKH LANGUAGE MORPHOLOGY

Turkish and Kazakh languages are Turkic languages which belong to Ural-Altaic language family. Since they are agglutinative languages, their words are formed from stems and recursive addition of proper suffixes to those stems. In order to create an efficient EBMT system for agglutinative languages, the usage of morphological processors for these languages is essential. The morphological analysis of Turkish is performed by a Turkish morphological processor developed earlier [68], [41] which uses morphology rules developed by Oflazer [42], and a morphological processor for Kazakh language [65] is developed using software Foma [66] to be used in our EBMT system. Affix types and grammatical naming in Kazakh morphological processor are also defined similarly to Turkish morphological processor [42], [41]. This chapter is organized as follows: First, quick review of finite state transducers and two level morphology is given. Then, similarities and differences of Kazakh and Turkish languages are mentioned. After, introduction of Kazakh Language morphology with details is given. Then, usage of morphological parses in our hybrid EBMT system are defined. It is essential for almost all agglutinative languages.

3.1. Finite State Transducers And Two Level Morphology

Finite state techniques are successfully applied to various areas of natural language processing such as the construction of lexical analyzers, parsing context free grammars and the compilation of morphological and phonological rules [69] [70].

Finite state transducers read their input symbol in defined state and give a corresponding output, then move to a new state. This improves the processing speed fundamentally. Practically, the processing speed is independent of the size of the rules.

We are interested in a finite state process for morphological analysis. Each word in a language is composed of a root and possible morphemes affixed to that root. The finite state transducer takes a word in and checks all possible roots and morphemes affixed to that root. Many previously determined rules work in parallel and they check the possibilities at any state. It may produce more than one output for each input string. It may also produce no output for a

given input, in which case it is said the input is rejected by the rules. For more compact information on finite state morphological analysis process, see [71], [72],[73].

Two-level morphology is a way of handling morphological structures by executing pseudo-parallel rules [72]. There are two levels of the system, *surface level* and *lexical level*. Surface level representation is the direct representation of input, as it is represented in the original language. Lexical level is the decomposed form of the input and is the output of the system when the surface representation is given as input. In a finite state transducer, normally the surface and lexical levels are represented as two expressions separated by a colon. For example, an expression like a:b is usually expected to mean “lexical form a is derived from the surface form b”.

Rules that denote the morphological modifications and variations are all executed in parallel and all the rules work on the same input. If all of the rules accept the input, then the machine accepts the input. However, if the input is rejected by any of the rules, then the machine rejects the input directly. There are four different rule types in such a system:

a:b => LC _ RC : lexical a is always realized as surface b in left and right context LC_RC

a:b => LC _ RC : lexical a is realized as surface b in only left and right context LC_RC

a:b <=> LC _ RC : lexical a is always realized as surface b in this context and nowhere else.

a:b /<= LC _ RC : a lexical a is never realized as surface b in the given context.

The morphotactic rules are compiled to a finite state transducer and are joined with these rules. The system as a whole tries to locate the roots and possible following suffixes for a given surface form input. If the system at any stage cannot locate a valid suffix or it discovers a situation violating the alternationrules, it returns with no answer. For a detailed explanation of two-level morphology refer to [42]. In this work application of FST with two-level morphology to Kazakh language morphology is defined in next section.

3.2. Turkish Language Morphology

There are considerably many studies on Turkish language because it is distinctive than other generally known languages such as English etc. In our thesis we use the study which was worked out before for Turkish [59], therefore we do not give all details of language here. Instead we make a comparison of two languages and point out the similarities and differences of both. Then we will give a detailed analysis of Kazakh language which is the main scope of this chapter.

3.3. Similar and Distinctive Features of Turkish and Kazakh languages

Turkish and Kazakh are both member of same group of languages, which is the Altaic branch of the Ural-Altaic language family. Thus, both are agglutinative languages which generate words by joining morphemes together and each morpheme represents one unit of lingiustic function with its meaning, such as “plural”, ”past tense”, etc. Thus, many meanings can be load to a single word with this agglutinave feature. For example, ev-de-ki-ler-in↔үй-де-гі-лер-дің means “of those which are at home”

Basic word order of Turkish and Kazakh sentences is Subject-Object-Verb (SOV). However, both languages’ grammar allow all combinations of component orders. Shifting order of components does not change meaning of a sentence but only changes the stress. However, there are some parts of constituents which do not change freely such as noun phrases. For example, while *Sarı armut* (yellow pear) is legal, *armut sarı* (apple yellow) is not.

Turkish and Kazakh have no grammatical gender. Thus, nominal nouns do not have a gender as in Russian or Arabic languages. For example, in Russian every noun has its gender and all grammatical relations constructed according to these differences. дом окно страна (home, window, country); мой дом, мое окно, моя страна (my home, my window, my country);

In Turkish and Kazakh, the vowels of suffixes should have a harmony with the last vowel of that word. Vowel agreement in Turkish is more complicated than in Kazakh. For example, In Turkish there exist vowel harmony rules such as:

1. if the vowel of last syllable is **a** or **ı** then suffix will be added with vowel **a** or **ı**

2. if the vowel of last syllable is **o** or **u** then suffix will be added with vowel **u** or **a**
3. if the vowel of last syllable is **e** or **i** then suffix will be added with vowel **e** or **i**
4. if the vowel of last syllable is **ö** or **ü** then suffix will be added with vowel **e** or **ü**.

Here examples for each rule are as following:

1. baş-ım, baş-lar; dış-ım, dış-lar; (my head, heads; my exterior, exteriors)
2. toz-um, toz-lar; tur-um, tur-lar;(my dust, dusts;my tour, tours)
3. tez-im, tez-ler; diz-im, diz-ler; (my thesis, thesises; my knee, knees)
4. göz-üm, göz-ler; yüz-üm yüz-ler;(my eye, eyes; my face, faces)

However in Kazakh, there are only two rules related to vowel harmony of suffix, such as:

1. if the vowel of last syllable is **а, ы, о, у, ұ** then suffix will be added with vowel **а** or **ы**
2. if the vowel of last syllable is **ә, е, і, и, ө, ү** then suffix will be added with vowel **е** or **і**.

Implementation examples for each rule are given as follows:

1. бас-ым бас-тар; тыс-ым тыс-тар; от-ым от-тар; ту-ым ту-лар; тұз-ым тұз-дар;(my head, heads; my outer side, outer sides; my wood, woods; my flag, flags; my salt, salts)
2. әл-ім әл-дер; ел-ім ел-дер; ит-ім ит-тер; өт-ім өт-тер; түс-ім түс-тер;(my power, powers; my people, peoples; my dog, dogs; my gall, galls; my dream, dreams)

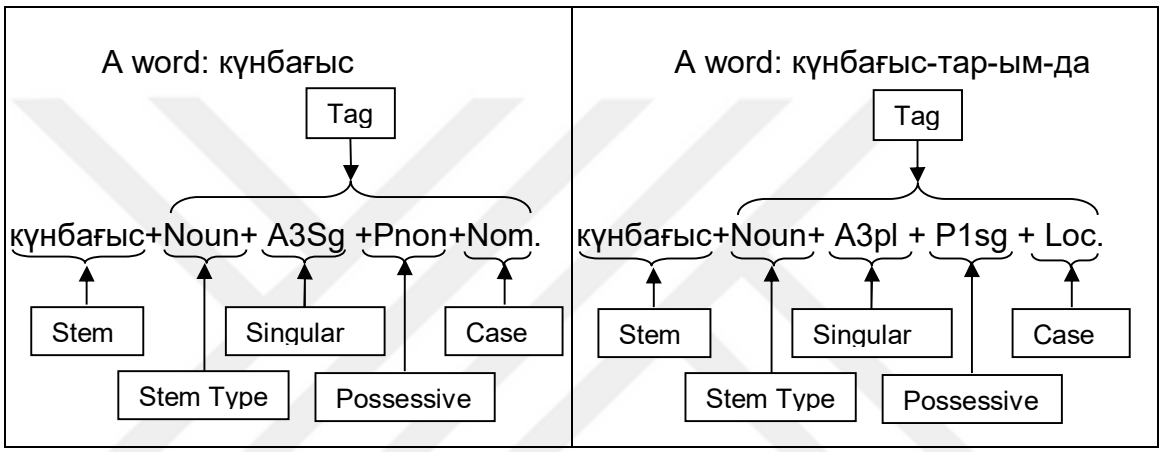
Differently in Kazakh there exist consonant harmony rules that are given in Section 3.4.1.

Inflectional and derivational morphotactics are nearly same with some implementation differences. Moreover, order of inflectional morphemes are also same in both languages. Thus, a noun first takes singular or plural (A3sg, A3pl)

suffixes, then possessive suffixes, after that case suffixes and at last derivation suffixes if they exist.

Every nominal root at least has a lexical form of Noun+A3Sg+Pnon+Nom. Therefore, a noun root күнбағыс which means “sunflower” has a morphological analysis as күнбағыс+Noun+A3Sg+Pnon+Nom. For comparison, an other inflected form of word күнбағыс which is күнбағыс-тар-ым-да means *in my sunflowers* and its morphological analysis is illustrated in Table 1.

Table 1. Example for Morphological Analysis of Noun and Suffix Order



Derivational rules are similar in both languages such as every noun can be derived to noun, adjective, adverb and verb. Moreover, every verb can be derived to noun, adverb, adjective, verb etc. For example,

күз, "summer"→күзгі, "of summer" (Derives from Noun to Adjective)

жүр→жүр-гелі, "since coming" (Derives from Verb to Adverb)

There are similar tenses of verbs in both languages, except some tenses which exist in Kazakh but not in Turkish or vice a versa.

There are 9 tenses in Turkish with modals and 14 composed tenses which have two tense suffixes, however in Kazakh there are 13 tenses with modals and 16 composed tenses which are constructed by compound verbs such as **екен** and **еді,** given in Table 2. Modals with asteriks (*) are not exact equivalent in Kazakh language but their usage without losing semantic meaning.

Table 2. Example for Morphological Analysis of Noun and Suffix Order

Tenses in Turkish	Example	Equivalent in Kazakh
1. Geniş Zaman *(Present simple tense)*	oku-**r** *(he reads)*	Болжамды Келер Шақ *(Future Indefinite Tense)*
2. Gelecek zaman *(Future tense)*	oku-y-acak *(he will read)*	Ауыспалы Келер Шақ *(Future Transitional Tense)*
3. Şimdiki zaman *(Present continuous tense)*	oku-yor *(he is reading)*	Нақ Осы Шақ (Auxilary Verbs) *(Present Definite Tense)*
4. Bilinen geçmiş zaman *(Definite past tense/Past tense)*	oku-du *(he read)*	Жедел Өткен Шақ *(Past Definite Tense)*
5. Öğrenilen geçmiş zaman *(Indefinite/Inferential past tense)*	oku-muş *(Presumably he read)*	Айғақсыз Бұрыңғы Өткен Шақ *(Past Narrative Indefinite Tense)* Айғақты Бұрыңғы Өткен Шақ *(Past Narrative Definite)* + екен
1. Emir Kipi *(Imperative modal)*	oku *(read)*	Бұйрық Рай *(Imperative Modal)*
2. Gereklilik Kipi *(Necessitative modal)*	oku-malı *(he have to read)*	-
3. İstek Kipi *(Wish modal)*	oku-y-a *(he would read)*	Қалау Рай *(Wish modal)*
4. Dilek Kipi *(Conditional modal)*	oku-sa *(if he reads)*	Шартты Рай *(Conditional modal)*
Composed Tenses		
a) Hikaye Kipi +dı		
6. Geniş *(Simple Past)*	oku-r-du *(he read)*	Болжамды Келер Шақ *(Future Indefinite Tense)*+ еді
7. Gelecek *(Future Past)*	okuyacak-tı *(he was going to read)*	Ауыспалы Өткен Шақ *(Past Transitional Tense)* + еді
8. Geçmiş *(Pluperfect Past)*	okumuştu *(Presumably he had read)*	Айғақсыз Бұрыңғы Өткен Шақ *(Past Narrative Indefinite Tense)*+ еді Айғақты Бұрыңғы Өткен Шақ *(Past Narrative Definite Tense)*+ екен
9. Gereklilik *(Necessitative Past)*	okumalıydı *(he had to read)*	-
10. Dilek *(Conditional Past)*	okusaydı *(if he had read)*	Шартты Рай *(Conditional modal)* + еді
b) Rivayet Kipi +mış		
11. Geniş *(Simple Inferential)*	okurmuş *(It seems he reads)*	Болжамды Келер Шақ *(Future Indefinite Tense)*+ екен
12. Gelecek *(Future Inferential)*	okuyacakmış *(It seems he shall read)*	Ауыспалы Өткен Шақ *(Past Transitional Tense)*+ екен
13. Şimdiki (Continuous Inferential)	okuyormuş *(It seems he is reading)*	Нақ Осы Шақ (Auxilary Verbs) *(Present Definite Tense)* + екен
14. Dilek (Conditional Inferential)	okusaymış *(it seems he should read)*	Шартты Рай *(Conditional modal)* + екен
c) Şart Kipi +sa		
15. Şimdiki *(Conditional*	okuyorsa	Нақ Осы Шақ (Auxilary Verbs) *(Present*

Continuous)	*(if he is reading)*	*Definite Tense)*+ca
16. Gelecek *(Conditional Future)*	okuyacaksa *(if he shall read)*	*Ауыспалы Өткен Шақ (Past Transitional Tense) +болса (*)*
17. Geniş *(Conditional Simple)*	okursa *(if he reads)*	*Болжамды Келер Шақ (Future Indefinite Tense)+болса(*)*
18. Geçmiş Bilinen *(Conditional Past)*	okuduysa *(if he read)*	*Айғақты Бұрыңғы Өткен Шақ (Past Narrative Definite Tense)+болса(*)*
19. Geçmiş Öğrenilen *(Conditional Inferential)*	okumuşsa *(if it seems he read)*	*Айғақты Бұрыңғы Өткен Шақ (Past Narrative Definite Tense)+болса(*)*
20. Gereklilik *(Conditional Necessitative)*	okumalıysa *(if he has to read)*	-

Differently in Kazakh there are auxilary or compound verbs which define physical position of teller or contributors to a story. In Kazakh language, there are 4 auxiliary verbs that are added to main verbs by indicating continuity and the physical position of the doer. These auxiliary verbs are **“тұр”** (standing by), **“жүр”** (walking), **“отыр”** (sitting) and **“жатыр”** (lying down). Thus, every verb in Kazakh language can be different in meaning depending on which auxiliary verb added to that verb. They are legal only on Present Definite tense (Нақ Осы Шақ) in Kazakh language. However after being added to the main verb, they can also act as a main verb and take all suffixes of all tenses.

3.4. Introduction To Kazakh Language Morphology

3.4.1. Vowel and Consonant Harmony

Kazakh is officially written in the Cyrillic alphabet. In its history, it was represented by Arabic, Latin and Cyrillic letters. Nowadays switching back to Latin alphabets in 20 years is planned by Kazakh government. In this thesis, the current Cyril version is used for convenience. Two main issues of language such as morphotactics and alternations can be dealt with finite-state tools. In our morphological analyzer, morphotactic rules are represented by encoding a finite-state network and a finitestate transducer for alternations is constructed using Foma finite-state tools. Then, the formed network and the transducer are composed into a single final network which cover all morphological aspects of the language such as morphemes, derivations, inflections, alternations and geminations [40].

Vowel harmony of Kazakh language obeys one rule such that vowels in each syllable should match according to being front or back vowel. It is called

synharmonism and it is basic linguistic structure of nearly all Turkic languages [74]. For example, the word қала-лар-дың (“of cities”) has the stem қа-ла (“city”) whose two syllables contain back vowels and all added suffixes should contain back vowels according to the vowel harmony rule. Both of its suffixes –лар and –дың contain back vowels. Here, –лар is an affix of plural form and –дың is an affix of genitive case. However, as stated before, there are a lot of loan words from Persian and generally they do not obey the vowel harmony rule. For example, мұ- ға-лім means “teacher”, and its first two syllables have back vowels and its last syllable has a front vowel. Since suffixes to be added are defined according to the last syllable, the vowel of the last syllable should match with all other remaining morphemes. For example, the word мұғалім-дер-дің (“of teachers”) whose last two syllables contain front vowels obeys the vowel harmony rule.

On the other hand, there are morphemes with static front vowels which are independently from the type of the last syllable can be added to all words such as instrumental suffix –мен which contains a front vowel. In this case, all suffixes that are added after that suffix should also contain front vowels. Words in Kazakh language take suffixes with vowels a or ы if their last syllables contain back vowels, and in other cases they take suffixes containing vowels e or i. The developed morphological analyzer provides mappings between lexical and surface level representations of Kazakh words. Although users only deal with lexical and surface level representations of the words, the morphological analyzer also uses intermediate representations of the words.

In order to construct a finite state transducer for alternation rules, some capital letters such as A, J, H, B, P, C, D, Q, K are defined in intermediate level and they are invisible by users. These representations are used for substitutions such as A is for a and e and J is for ы and i. So, if the suffix дA should be added to a word according to morphotactic rules, it means that actual suffixes да or де are considered according to alternation rules. There are groups of letters that are defined according to their sounds and these groups are used in alternation rules .

Consonant harmony rules are varied according to last letters of words in morphotactic rules. As in Figure 5, different patterns are presented in order to

visualize the relation between common valid rules and to generalize morphotactic rules. Consonants in Figure 5 are separated into three groups such as sonorous, voiced and unvoiced consonants. Sonorous and voiced consonants are also grouped as Type 1 and Type 2. In Figure 5, Type 2 unvoiced consonants and unvoiced consonants have same pattern and this means that similar suffixes are added after them. Thus, Figure 5 defines five different patterns which affect suffix types to be added to words according to morphotactic rules.

Name	Type 1	Type 2
Sonorous Consonant	л р у й	м н ң
Voiced Consonant	з ж	б в г ғ
Unvoiced Consonant	п ф қ к т с ш щ х ц	
Vowel	а е ә і ы ө о ұ ү и	

Figure 5. Groups of Kazakh letters according to their

All rules for suffixes depend on last letters of morphemes in morphotactic rules. Figure 6 and Figure 7 give some groupings that can be made in order to set some generalized rules overall. Patterns of last letters of morphemes in Figure 6 and Figure 7 are matched with groups of letters presented in Figure 5. For example, locative case affix is –дА, if the last letter is vowel, sonorous consonant or voiced consonant of Type 1. It is –тА, if the last letter is unvoiced consonant or voiced consonant of Type 2. It is –ндА, if the last letter is ы or і, since a word is on third personal possessive state. Here A is for a or e according to the last syllable of containing front or back vowel. So visually some cases have similar patterns and some are exactly same [75]. Here boxes presented by numbers such as 1, 2 and 3 are for personal possessive agreements.

GROUP 1												
Ablative Case					Locative case					Dative Case		
дАн					дА					ТА				
тАн					тА					ТА				
нАн				3	ндА				3	нА				3
										А				1/2

Figure 6. First Group of Similar Alternation Rules according

GROUP 2												
Genitive Case					Accusative case					Poss. Affix-2				
дЈң					дЈ					дікі				
тЈң					тЈ					тікі				
нЈң				3	нЈ					нікі				
					н				3					

Figure 7. Second Group of Similar Alternation Rules according

For example, word әке (“father”) in ablative case with none possessive agreement will take suffix –ден, but in third person possessive agreement it takes suffix –нен. Thus әке+Noun+A3Sg+Pnon+Abl→әке–ден (“from father”) and әке+Noun+A3Sg+P3Sg+Abl→ әке–сі–нен (“from his father”) mappings occur. This is different from words which end with vowels. For example, a word сіңлі means “little sister” and its ablative case is analyzed as сіңлі+Noun+A3Sg+Pnon+Abl→сіңліден.

According to those similarities there are generalized rules which are valid in many cases in grammar including verbs and derivations. In Figure 6, locative and dative suffix rules are nearly identical which can be observed visually. Also, accusative and possessive pronouns of Type 2 are same. In dative case, if the last letter is a vowel and the last syllable contains a back vowel then T is replaced by ғ or г. Also, if the last letter is an unvoiced consonant and the last syllable contains a front vowel then T is for қ or к. Thus, the word бала (“child”) becomes бала–ға (“to child”) and the word әке (“father”) will be әке–ге (“to father”) in ablative case. The reader can observe that the last letter is a vowel, at the same time it is a front vowel in the last syllable and thus T→ғ mapping occurs. Also, the last letter of the word кітап–қа (“to book”) is an unvoiced

consonant and its last syllable contains a back vowel, thus Т→қ mapping occurs. The last letter of the word мектепке (“to school”) is an unvoiced consonant and its last syllable contains a front vowel, thus Т→к mapping occurs. After detailed analysis of the language it can be seen that there are mainly common rules of alternations valid over all grammar. There are about 57 main alternation rules defined for all system together with generalized rules and 13 exception rules for each case separately. All these rules are implemented with Foma finite-state tools, and they are defined and composed in a Foma file [66]. For instance, some of most common alternation rules are given below and they are called by capital letters defined in intermediate level and they are not available on surface level. As mentioned before they are invisible by users. They are represented by surface level characters or they drop. In the following rules, 0 stands for empty string.

Rule H & Rule B: H is realized as 0 or J, B is realized as 0 or A.

[H → 0, B → 0 || [V owel]%+_[Cons]] [H− > J, B− > A]

If the last letter of the morpheme is a vowel and the first letter of the suffix is a consonant then H and B are realized as 0. Otherwise, they are realized as J and B, respectively. Some examples of Rule H and Rule B are as follows, and two of examples also uses Rule J and Rule A.

Rule H: ана-Нм→ана-м, “my mother”

Rule H & Rule J: іш-Нм→іш-Jм→іш-ім, “my stomache”

Rule B: ere-Bp→ere-p, “will sharpen”

Rule B & Rule A: бар-Вр→бар-Ар→бар-ар, ”will go”

Rule J & Rule A: J is realized as ы or i and A is realized as й, a or e.

[A → й || [Vowel]%+_] [A → a, J → ы || [BVowel](Cons)∗%+?∗_] [A → e, J → i || [FVowel](Cons)∗%+?∗_]

If the last letter of morpheme is a vowel then A is realized as й, and if the last syllable of a morpheme contains a back vowel then A and J are realized as a and ы. Otherwise, if the last syllable of a morpheme contains a front vowel then A and J are realized as e and i. Some examples of Rule R and Rule A are as follows.

бас-Нм→бас-Jм→басым, “my head”

дос-тАр→дос-тар, “friends”

дәптер-лАр→дәптер-лер, “copybooks”

барма-Амын→барма-ймын, “I will not go”

Rule T: T is realized as қ, ғ, к or г depending on previous characters.

[T → қ || [BV owel](?)[UV Cons]%+_]

[T → к || [FV owel](?)[UV Cons]%+_]

[T → ғ || [BV owel](?)[0|SCons|V Cons1]%+_]

[T → г || [FV owel](?)[0|SCons|V Cons1]%+_]

This rule is illustrated partly in Table 2 for dative case. It is one of generalized rules which are valid in many cases such as derivation of nouns, adjectives and verbs. Some examples of Rule T are as follows.

бала-Та→бала-ға, "to child" (Noun in Dative)

жаз-Ты→жаз-ғы, "of summer" (Adjective)

жүр-Телі→жүр-гелі, "since coming" (Verb)

есті-т-Тіз→есті-т-кіз, "make hear"(Causative Verb)

Orthographic rules are given in Appendix 1.

3.4.2. Nouns

Nouns in Kazakh language take singular or plural (A3sg, A3pl) suffixes, possessive suffixes, case suffixes and derivation suffixes. In addition, nouns can take personal agreement suffixes when they are derived into verbs. For example, кітап-тар-да-ғы-лар-дың which means “of those which are in books” has the following morphological analysis. кітап+Noun+A3Pl+Pnon+Loc ^DB+Noun+Zero+A3Pl+Pnon+Gen. Every nominal root at least has a lexical form of Noun+Sg+Pnon+Nom. Therefore, a noun root кітап which means “book” has a morphological analysis as кітап+Noun+A3Sg+Pnon+Nom. These inflections of noun are given in FST diagram in Figure 8. Part of lexicon is given in Appendix 2. It can be seen that a nominal root can be in singular form by adding (+0) no suffix which is in fact third personal singular agreement (A3sg)

and by adding suffix

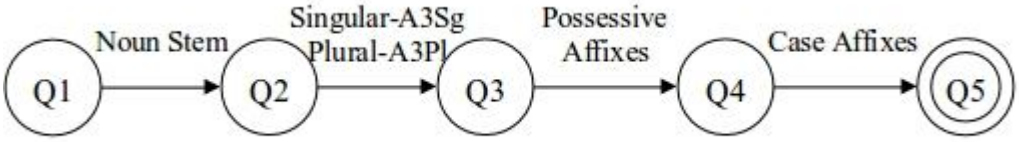


Figure 8. The FSA model of inflectional changes of a

(+PAr) in plural form which is in fact third personal plural agreement (A3pl). Here P is an intermediate level representation letter for letters д, т or л in surface level. After, possessive affixes (+Pnon:0, +P1sg:Нм, +P2sg:Нң, +P2psg:НңJз, +P3sg:cJ, +P1pl:НмJз, +P2pl:Нң, +P2ppl:НңJз, +P3pl:cJ) and case affixes (Nom, Dat, Abl, Loc, Acc, Gen, Ins) are added. Here H and J are intermediate letters. All morphotactic rules together with adjective, pronoun, adverb and numerals are given in Figure 9. It can be observed that every adjective can be derived to noun and nouns with relative affix can be derived to adjectives. There are other derivations which are produced by adding some specific suffixes between verbs and nouns, adjectives and adverbs, adjectives and nouns. In order to get rid of complex view those derivations are not explicitly shown in Figure 9. In our morphological analysis system, root of word is a starting point for morphemes defined in lexicon file, and other morphemes are added according to morphotactic rules. All possible morphemes of Kazakh language are defined in the lexicon of the morphological analyzer.

In order to give better comprehension of morphotactic rules illustrated in Figure 9, let us clarify Pronoun part. There are seven kinds of pronoun types in Kazakh language[М. Ә Қараев Қазақ тілі] such as Personal, Reflexive, Indicative, Interrogative, Definite and Indefinite pronouns.

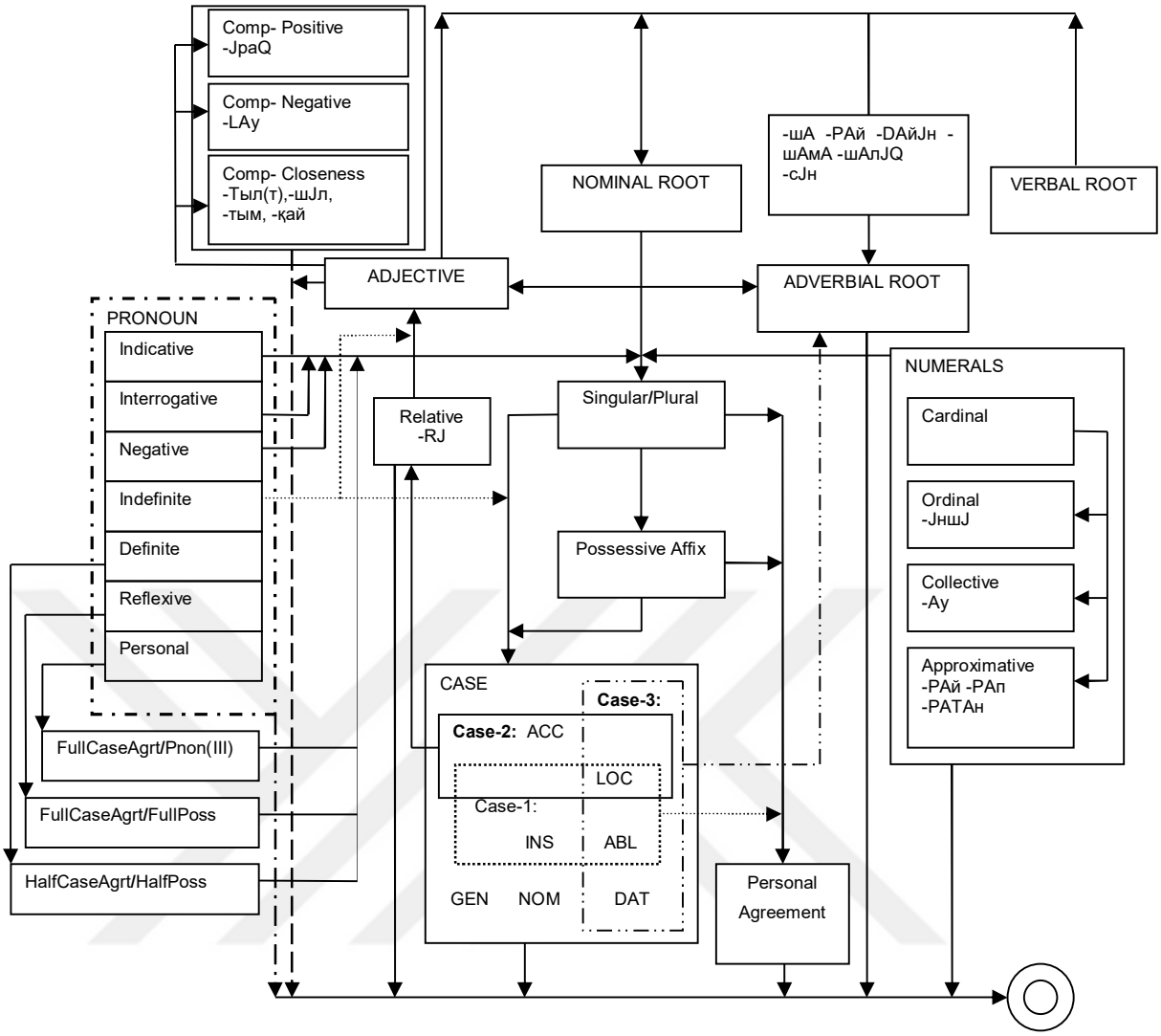


Figure 9. Morphotactic Rules for Nominal Roots

сіз(You polite), ол(He\She), біз(We), сендер(You plural), сіздер(you polite plural) and олар(They). Only theird person pronoun has its possessive agreement. Thus, in Figure 9 Personal pronoun is indicated as FullCaseAgr with Pnon except 3rd person pronoun.

All case agreements of personal pronoun are examplified in Table 3. Personal Pronoun has full case agreement, but they do not have possessive agreement except 3rd person pronoun such as ол- оны, оның, оныңыз, онысы. On the other hand, some of definite pronouns have case agreement and some are not. Also, not all of definite pronouns have possessive agreements. For this reason, it is indicated to have half case agreement and half possessive agreement as *HalfCaseAgrt/HalfPoss*.

Table 3. Personal Pronoun and its Case Agreement

Pers. Pronoun \ Case	Nom.	Abl.	Loc.	Dat.	Gen.	Acc.	Ins.
Мен (I)	Мен	менен	менде	маған	менің	мені	менімен
Сен(You)	Сен	сенен	сенде	маған	сенің	сені	сенімен
Сіз(You p)	Сіз	сізден	сізде	сізге	сіздің	сізді	сізбен
Ол(He\She)	Ол	одан	ода	оған	оның	оны	онымен
Біз(We)	Біз	бізден	бізде	бізге	біздің	бізді	бізбен
Сендер(You pl)	Сендер	сендерден	сендерде	сендерге	сендердің	сендерді	сендермен
Сіздер(You ppl)	Сіздер	сіздерден	сіздерде	сіздерге	сіздердің	сіздерді	сіздермен
Олар(They)	Олар	олардан	оларда	оларға	олардың	оларды	олармен

On tte other hand, Indicative pronouns usually encountered as an adjective form in a sentence and can not have its plural form, case and possessive agreements, if not used as object. Indefinite pronouns can not have possesive agreements in its noun form in a sentence, if not used as an adjective. Moreover some nominal roots convert to adverbial roots by adding proper affixes such as *–шA, -PAй, -DAйJн, -шAмA, –шAлJQ, -cJн.* Here letters defined in latin alphabet are intermediate letters which encountered for several letters in cyrillic alphabet thus in Kazakh language. For example, қазақ-ша(in Kazakh), ақша-лай(in cash), бота-дайын(as brood of camel), сон-шалық(as so much), осын-шама(as so many) and кешкі-сін(in evening time).

For cases, there are three different groups of case agreements. Locative, Instrumental and Ablative cases are in group of *Case-1* and nouns in this group can be ended with Personal agreement affixes. For example, жүрег-ім-мен-сің which means “you are with my heart”. Accusative and Locative form are in *Case-2* group and noun which is in this group can be turn to adjective by adding proper suffix *-RJ* or can be derived back to noun. R is an intermediate letter which is encountered for “ғ” and “г”. For example, үй-де-**гі**-лер-дің which means “of those which are at home” has the following morphological analysis үй+Noun+A3sg+Pnon+Loc^DB+Noun+Zero+A3Pl+Pnon+Gen. Here a suffix “-гі” derives noun “үй” which is in locative case to noun that can have its plural form, case and possessive agreements. Locative , Ablative and Dative are in Case-3 group and nouns in this group can be treated as an adverbs in a sentence.

3.4.3. Verbs

Verbs are terms which define actions and states. Mainly three tenses exist such as present, future and past as stated in Figure 11. Moreover, conditional, optative and imperative moods are also defined. However in detailed form there are thirteen tenses together with modals in Kazakh language. These tenses are worked out from many resources where presentation and naming have variance among each other according to their scholars [76], [77], [78]. For example, according to Isaeva and Nurkina [78] Ауыспалы Келер Шақ, “Future Transitional Tense” denotes action in future and has same affix as Present Tense. However, Mamanov [76] points out that Ауыспалы Келер Шақ, “Future Transitional Tense” denotes present action. Our work is mainly based on morphology of Kazakh language defined by Karaev in [79]. Tenses in Kazakh language and their detailed properties are given in Appendices 6, 7, 8. Additionally, there are large amount of auxiliary verbs which define tenses and some modal verbs. However, in cases that auxiliary verbs are not used as verbs, they become adverbial adverbs or participles which define verb or noun . In Figure 12, morphotactic rules of verbs and modals are given.

	Шақ/Tense	Жұрнақ/Suffix	
Future - Келер	Болжамды Келер Шақ *Future Indefinite Tense*	ар/ер/р	
	Мақсатты Келер Шақ *Future Goal Oriented Tense*	бақ/бек пақ/пек мақ/мек	
	Ауыспалы Келер (Осы) Шақ *Future (Present) Transitional Tense*	а/е/й	*Present*- Осы
	Мақсатты Осы Шақ *Present Goal Oriented Tense*	қалы/ғалы келі/гелі	
	Нақ Осы Шақ *Present Definite Tense*	ып/іп/п	
	Дәл Осы Шақ *Present Progressive*	да/де	
Past - Өткен	Жедел Өткен Шақ *Past Definite Tense*	ды/ді ты/ті	
	Ауыспалы Өткен Шақ Past *Transitional Tense*	атын/етін йтын/йтін	
	Айғақты Бұрынғы Өткен Шақ *Past Narrative Definite Tense*	ған/ген қан/кен	
	Айғақсыз Бұрынғы Өткен Шақ *Past Narrative Indefinite Tense*	ып/іп/п	

Figure 11. Tenses of Verbs in Kazakh Language

Derivations of verbs to nouns and adverbs with specific suffixes are shown with asterisk in Figure 12. Verbs can be in reflexive, passive, collective and causative forms. For instance, verb тара-у which means "to comb" is represented as тара-н-у in reflexive infinity form, тара-л-у in passive infinity form, тара-с-у in collective infinity and тара-тQJз-у and тара-тTJр-у in causative infinity form. Here, Q, J and T are intermediate letters. However not all verbs can have all of these forms at the same time. Verbs in infinity form are generally formed with last letter y, and the verb келу which means “to come” is an example for this case. The system is performing over generalization on verbs which take auxiliary verbs on appropriate tenses. Those verbs are analyzed as derived adverbs or incomplete verbs on that tense since every verb of a sentence should have a personal agreement and the personal agreement affix is added to the verb itself after the tense suffix or to the auxiliary verb. Some of the tenses have different personal agreement endings and they are presented in Table 4 and in Table 5.

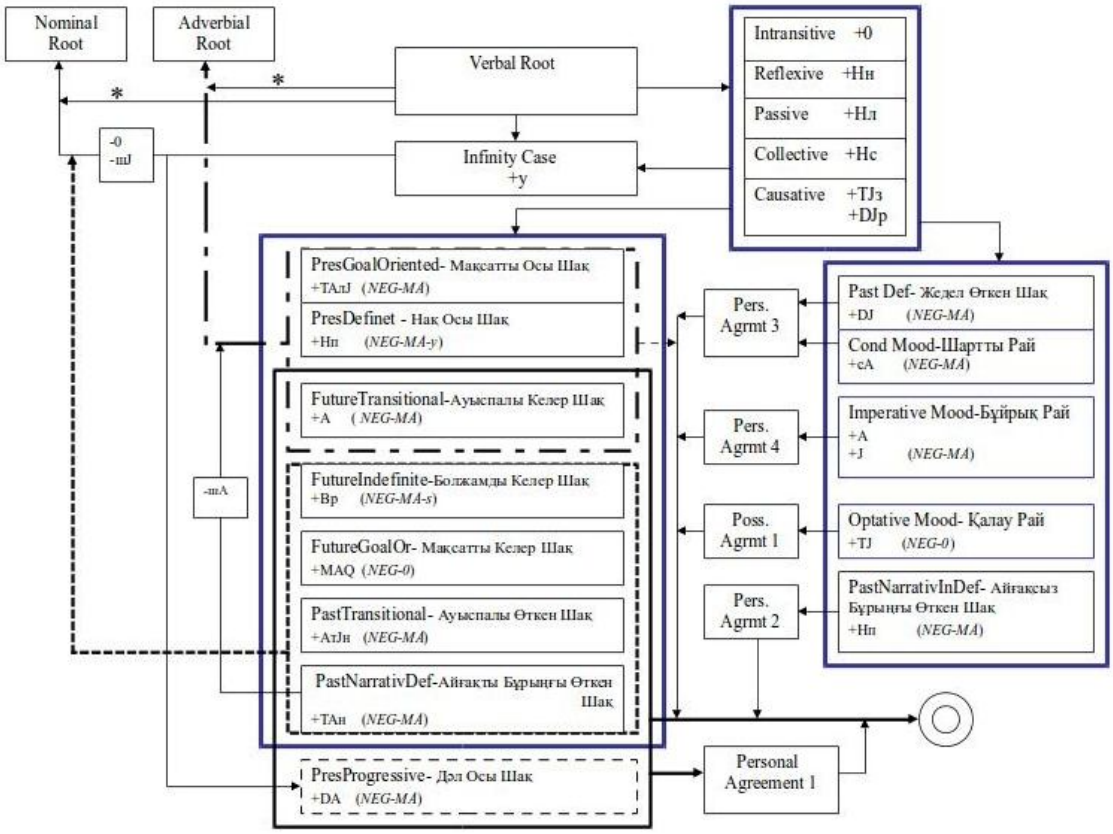


Figure 12. Morphotactic Rules of Verbs in Kazakh Language

All agreement are added as suffix to related verb, thus their orthography are differentiate according to the last characters of a verb. For this reason, all agreements given in Table 4 and Table 5 are shown in intermediate level in order to be varied dependent on last characters and their equivalent are given in paranthesis.

Table 4. Types of Personal Agreements

Personal Pronoun			**Agrmt 1**	**Agrmt 2**	**Agrmt 3**	**Agrmt 4**
мен (I)	P1Sg	Verb+	MJн	MJн	м	йJн (a,e)+
біз (we)	P1Pl		MJз	MJз	Q	йJQ
сен (you)	P2Sg		cJң	cJң	ң	-
сендер (you *)	P2Pl		cJңдAр	cJңдAр	ңдAр	ңдAр
сіз (you+)	P2PSg		cJз	cJз	ңJз	ңJз (J)+
сіздер (you+*)	P2PPl		cJздAр	cJздAр	ңJздAр	ңJздAр
ол (he,she)	P3Sg		-	DJ	-	cJн
олар (they)	P3Pl					
(M→м,п,б), (J→ы, і), (Q→қ, к), (A→a,e), (D→д,т)						

Table 5. Possessive Pronouns

Possessive Pronoun 1			
менің(My)	VERB + GJ	м	келу (to come)
біздің(Our)		мJз	
сенің(Your)		ң	
сендердің(Your*)		лApJң	
сіздің(Your+)		ңJз	
сіздердің(Your*+)	+ QJ	лApJңJз	
оның(his,her)		cJ	
олардың(their)		лApJ	
(Q→қ, к), (A→a,e), (J→ы, і), (G→ғ, г)			

* 2nd Person Plural
+ 2nd Person Polite
*+ 2nd Person Plural Polite

In the constructed morphological analyzer, we make the analysis of every single word and for that reason generalization of some rules are made by giving more than one result. Thus compound verbs are examined separately. For example, кел-гелі тұр-мын which means “I am planning to come” is an example of this usage. Here тұр is an auxiliary verb which actually defines the tense of the verb and takes a personal agreement affix. Without an auxiliary verb, the word кел-гелі means “since coming” and it is derived as an adverb. Thus, in order to choose a correct one we developed the disambiguation system which is explained in next chapter.

3.5. Morphological parse of a word

A morphological parse of a word in Kazakh can contain derivational and inflectional suffixes same as Turkish. For example, the morphological parse of етікші (shoemaker) is:

Here етік (shoe) is the stem and the rest of the morphological parse is the whole tag of the word. In this parse, ^DB shows a derivation boundary and Agt is the name of the derivational suffix. In a derived word, the collection of final morphemes after the last derivation ignoring the derivational suffix name is called as final tag and in our example it is Noun+A3sg+Pnon+Nom where Agt is eliminated. The final tag of an underived word is its whole tag. For example, the morphological parse of the word етіктің (of boot) is етік+Noun+A3Sg+Pnon+Gen, and Noun+A3Sg+Pnon+Gen is both its final tag and whole tag. Turkish words have similar morphological structures, and final and whole tags of Turkish morphological parses are found similarly [59]. In Chapter 7, we define statistical language model according to tags and stems of words.

3.6. Evaluation

In this section evaluation for morphological processor is presented. As mentioned before, the morphological processor is implemented using Foma

finite state tools. Morphotactic rules and possible morphemes are defined in the lexicon file. Alternation rules of Kazakh language are defined and the rules are composed with the lexicon file in a Foma file. Some loan words, proper names and technical terms are not included. The system is working in two directions as at lexical and surface level. Due to the ambiguities in language there is no one-to-one mapping between surface and lexical forms of words and the system can produce more than one result. There are approximately 15000 words in our test corpus which are selected from the web [80]. The percentage of correctly analyzed words is approximately %99. In the lexicon of the morphological analyzer, there are 3709 verbs, 13149 nouns, 3047 adjectives, 1218 adverbs, 794 conjunctions and 100 postpositions and numerals are included. The errors of the morphological analyzer are mainly the errors that appear in the analysis of technical, abbreviated and loan words which do not obey alternation rules of Kazakh language. The system is tested with four files in our test corpus and their results are given in Table 6. The files 1.txt and 2.txt have less such words than the files 3.txt and 4.txt. For example, the word фактілер which means "facts" is not correctly analyzed and it is derived from a loan word. Since it is a loan word, it doesn't obey Kazakh language rules.

Table 6. Results of morphological analysis

Files	Total Words	Correct	Uncorrect	Average Parse Per Word	Precision
1.txt	6462	6432	30	7.09	0.995
2.txt	3124	3093	31	6.91	0.990
3.txt	2836	2784	52	7.11	0.982
4.txt	2532	2493	39	6.65	0.985
Total	14954	14802	152	6.98	0.990

4. MORPHOLOGICAL DISAMBIGUATION

Natural language is complex due to its being natural and having mental influence of a speaker with effects of cultural, social, historical and geographical background of his society. Independent from the context where it is used, a word in a natural language can have more than one meaning. This case is called the ambiguity of a word and it is a big issue to be considered for any natural language processing task with even well-defined grammar rules. Especially this ambiguity problem is more complex for agglutinative languages. Kazakh language is an agglutinative language in which every affix converts a given word to a different form. Thus, its morphological disambiguation process is harder than others because it has more morphological parses for words. Statistical methods [51] [52]create probabilistic models from morphologically tagged texts and use these models to disambiguate words by selecting most probable morphological parses. There is a statistical morphological analysis for Turkish worked out using n-gram models for inflectional and final tags of words [53]. Rule-based morphological disambiguators [54] [55] [56] use hand-crafted rules to select correct parses of words or eliminate some of illegal parses of words. Disambiguation rules can be also learned from tagged texts using transformation-based learning approaches [57]. Hybrid systems [58] use both statistical 2 knowledge and disambiguation rules in disambiguation process. Turkish morphological disambiguator developed[59] by Kutlu and Cicekli uses both transformation based approach and rule-based approach. The morphological disambiguator for Kazakh language described in this thesis use transformation-based approach and it is a variation of Brill tagger[57].

4.1. Brill Tagger

Part of Speech tagger suggested by Eric Brill in 1995 is error driven transformation-based tagger. It is a supervised learning, which tends to minimize error. Also, it is transformation-based since a set of learned rules defines a tag to be assigned to each word and changes it. Thus, it first assigns the most frequent tag to known words. If the word is unknown, it just assigns the tag "noun" to it. However, it is obvious that everything which is unknown can not be a “noun”. So, if it is an error it should be minimized. Then, successively applying these rules by changing the incorrect tags to approach higher precision

gives a quite high accuracy. Thus, Brill Tagger can be briefly summarized as an error driven transformation-based tagger method which aims to minimize the total error. Our disambiguation system which is a variation of Brill Tagger is based on the idea of Kutlu and Cicekli [59], which was constructed for Turkish language earlier. Our system consists of two main parts such as training and disambiguation. First of all, we created a corpus for morphological disambiguation and words of this corpus are analyzed using our morphological analyzer [65]. The correct morphological parses of words are manually tagged. As a result, we obtained a manually tagged corpus and it is used for training and validation purpose. In Figure 13, general view of training and diambuguating corpus is given.

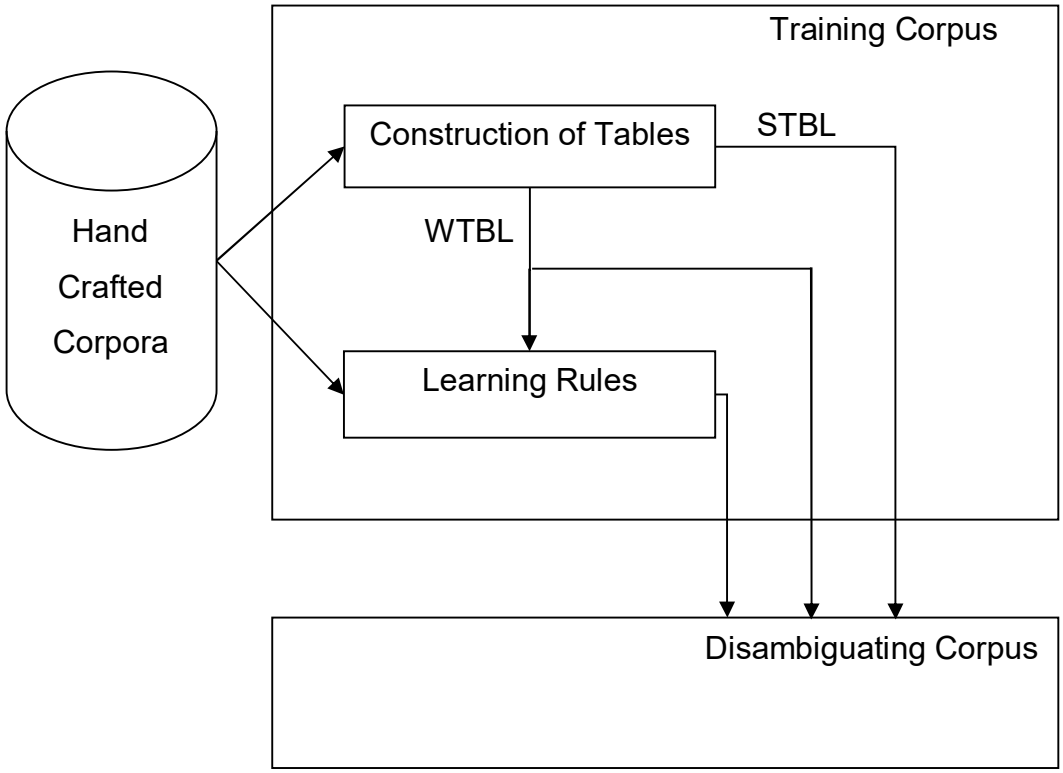


Figure 13. General View of Training and Disambiguation Corpus

4.2. Training

The training corpus is used to construct tables such as Most Likely Tag of Word Table (WTBL) and Most Likely Tag of Suffix Table (STBL). All morphological parse frequencies of words are kept in the table (WTBL) and all morphological

parse frequencies of suffixes are present in the table (STBL) in sorted order. It means that the first tag for a word or a suffix has the biggest frequency and thus is the most likely tag in each case. A part of WTBL is presented in Table 7, where it can be observed that a word “бұл” means *this* appears 224 times in all, 179 times as indicative pronoun and 44 times as noun and 1 times as conjugate.

Table 7. A part of WTBL

бұл:224	
Pron+IndcP+A3Sg+Pnon+Nom:	179
Noun+A3Sg+Pnon+Nom :	44
Conj:	1
Noun+A3Sg+Pnon+Nom^DB+Verb+Zero+Pres+A3Sg:	0
Noun+A3Sg+Pnon+Nom^DB+Adj+Zero: 0	

Table 8. A part of STBL

лық:376	
Verb+Pos^DB+Adverb+AfterDoningSo:	45
Noun+A3Sg+Pnon+Nom^DB+Adj+FitFor:	39
Noun+A3Sg+Pnon+Nom^DB+Noun+Theme+A3Sg+Pnon+Nom:	27
Verb^DB+Verb+Pass1+Pos^DB+Adverb+AfterDoningSo:	18
Noun+A3Sg+Pnon+Nom^DB+Noun+Theme+A3Sg+Pnon+Nom^DB+Adj+Zero:	5
Noun+A3Sg+Pnon+Nom^DB+Noun+Theme+A3Sg+Pnon+Nom^DB+Verb+Zero+Pres+A3Sg: 3	
…	

Also a part of STBL for suffix “**лық**” is visualized in Table 8. Here morphological parse or tag of a word is taken as whole tag of a word. As mentioned before, if a word doesn’t have any derivation boundary its whole tag is its final tag. At this stage, disambiguation rules are induced by using table WTBL. In our disambiguation system, the induced possible rules are in the following 3 forms.

- Type1: Select TAG_A for $WORD_N$ if the tag of $WORD_{N-1}$ is TAG_B .
- Type2: Select TAG_A for $WORD_N$ if the tag of $WORD_{N-1}$ is TAG_B and if the tag of $WORD_{N+1}$ is TAG_C
- Type3: Select TAG_A for $WORD_N$ if the tag of $WORD_{N+1}$ is TAG_C , where TAG_A, TAG_B and TAG_C are possible tags from WTBL.

Here "Select TAG_A for $WORD_N$ if Condition" means that we select the morphological parse with TAG_A for $WORD_N$ if "Condition" is satisfied and TAG_A

is the tag of at least one of the morphological parses of $WORD_N$. If there is more than one morphological parses with TAG_A which belongs to that word, select the one with the highest frequency. If $WORD_N$ does not have a morphological parse with TAG_A, the rule does not have any effect on $WORD_N$. After all possible rules are found, each rule is tried in order to select the rule that gives the best precision increase. Here the precision value is evaluated as follows in 4.1:

$$Precision = \frac{Number\ of\ Correctly\ Tagged\ Words}{Number\ of\ Totall\ Words} \quad (4.20)$$

where Number of Correctly Tagged Words is the number of correctly tagged words in the data set (here the data set is the training set with most likely morphological parses), and Number of Total Words is the number of the words in the data set. After applying the selected rule, we repeat the process until there is no progress or the improvement after the last found best precision. All learned rules are kept in their learning order.

4.3. Disambiguating

All learned rules which are kept in their learning order are used in disambiguating process. Also, WTBL and STBL are used for disambiguation of a word. Thus the disambiguation system consists of five major procedures such as:

- Assigning of the Most Likely Tag of Word
- Assigning of the Most Likely Tag of Suffix
- Assigning of Tag according to possible Suffix
- Assigning with Fall-Back Heuristics
- Application of the Learned Rules

The system looks for the correct morphological parses applying the above procedures in the given order. The correct parse of a word can be selected after ambiguous words can be clarified by eliminating some invalid parses. Certainly, we can not have all words in our training corpus and words can be still ambiguous after some steps have been applied.

Differently from the disambiguation system [11], if word is unknown we try to find a word by chunking a word from the last letter to find valid previously

learned suffixes. For example, assume a word “сатып” which means "selling" is ambiguous. We look for the last letter, which is ”п” as suffix and the rest word, which is ”саты” as a stem. If we have such predefined suffix in STBL, we will take all most frequent parses. On other side, we look at a stem in WTBL. We are continuing this process until a stem with one letter is left. In this case, the fourth step which is "Selection with Fall-Back Heuristics" will force the system definitely select a parse for each ambiguous word.

There is a possibility of being a word without any predefined suffix and then we add this word as a noun, adjective derived from noun, verb derived from noun and adverb derived from adjective. After the system forces a word to be defined we will apply previously learned rules to choose the correct one. In Figure 11, there is a flow chart of disambiguation process.

As mentioned before the order of learning rules specifies its’ application order. Then, the system decide if a rule is applicable to a word or not on existent conditions. Thus, if all of the following conditions are satisfied, then a rule is accepted to be applicable to a target word. These conditions are:

- All aspects of a rule should be completely disambiguated and satisfied.
- The target word satisfies its condition and to be disambiguated or having at least one parse which accomplishes its condition.
- The target word contains at least one of the possible parses of the correct tag given in the selection part of the rule.

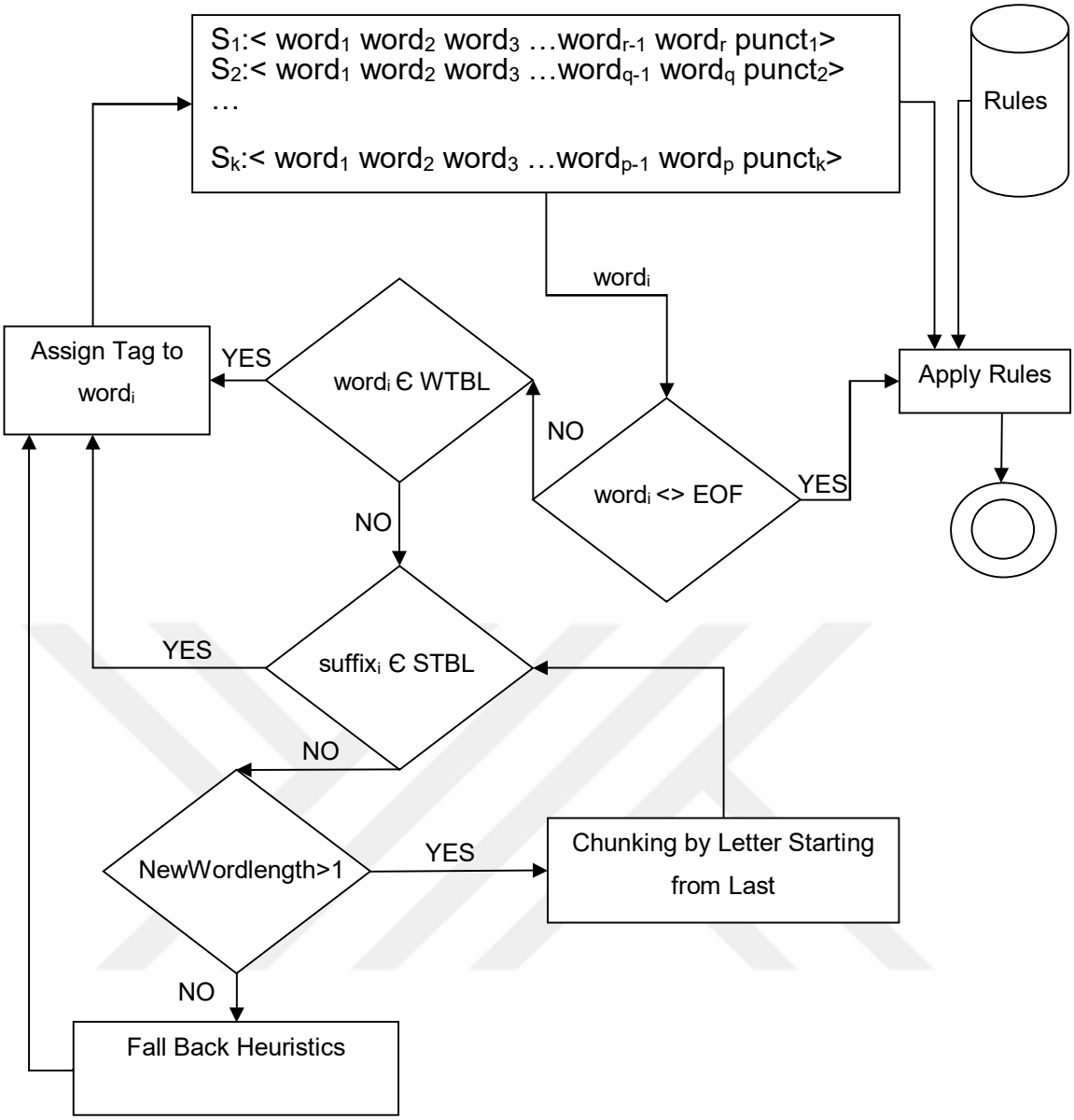


Figure 14. Flow Chart of Disambiguation Process

If the target word contains only one morphological parse satisfying the correct tag it is disambiguated. If the target word can not be completely disambiguated then at least some of its parses are selected as its possible parses for the next step. For instance, the following rule is applicable with its given aspects:

if final TAG of word$_i$ = *Adjective* ***and***

whole TAG of word$_{i-1}$=*Verb+Pos+Pres+A3sg*

then select MP containing whole tag *Adverb* for word$_i$

If the whole tag of the chosen MP of word$_{i-1}$ is *Verb+Pos+Pres+A3sg*, the final tag of at least one of possible MPs for word$_i$ is Adjective and word$_i$ consists at

least one possible MP having the whole tag Adverb, those MPs are assigned by this rule for $word_i$.

4.4. Evaluation

For the morphological disambiguator, a training corpus with 30171 words is used and all words in this training corpus are manually tagged with their correct morphological parses.

Table 9. Results of Morphological Disambiguation

		Correctly Disambiguated		
Files	Total Words	Before rules Applied	After Rules Applied	Precision
1.txt	6462	4588	5621	0.870
2.txt	3124	2249	2749	0.880
3.txt	2836	1956	2412	0.851
4.txt	2532	1774	2177	0.860
Total	14954	10567	12959	0.867

From this training corpus, our morphological disambiguator learned 512 disambiguation rules which is partly given in Appendix 3. The corpus used for the morphological analyzer is used a test corpus for our morphological disambiguator. This test corpus contains four files and 14,954 words in total. The results of disambiguated files are given in Table 9. 12,959 words of the test corpus with 14,954 words are correctly disambiguated. Without using the learned rules, 10,567 words are disambiguated just using most likely tags of words. Thus, 2,392 words are corrected by learned rules. The precision value for our morphological disambiguator is 0.87 percent. The accuracy can be raised by adding hand crafted rules to the disambiguation system.

5. LEARNING TRANSLATION TEMPLATES

As mentioned earlier, source level sentences in translation examples are morphologically analyzed and disambiguated before translation template learning process in order to obtain translation examples at lexical level. The translation template learning module infers translation templates from these examples at lexical level. The learned templates can be **atomic** templates without any variable or **general** templates with variables. Meanwhile, the system can also learn morphological types of variables in general templates with the help of morphology type lattices which are fed into the system for both languages. A morphology type lattice is a lattice for root words and morphemes of a language. At the bottom of the lattice, there are root words and morphemes, and at the top of the lattice is the main type ANY [63]. All other types in the lattice are subtypes of the main type. Also confidence factors of learned translation template are also calculated in the learning process [37]. Appropriate learned translation templates are chosen according to a source language sentence to be translated in order to translate that sentence.

Example based machine translation is a translation technique that can be modeled as if a man given sample sentences in source language (L1) with their correspondences in target language (L2), by memorizing these aligned pairs and seeing patterns in these translation examples, he becomes able to translate new sentences. Aligned bilingual translation pairs are given to our EBMT system in order to learn translation templates. The induction of a translation template is performed by replacing differing parts in given aligned examples by variables. Replacing a difference part with variables is called as the generalization of those differing parts.

In order to find differing parts for each pair of translation examples, a match sequence is generated for them. A match sequence for a pair of translation examples consists of two match sequences where one of them is a match sequence for source language sentences in the pair and the other one is the match sequence for target language sentences. A match sequence of two sentences is a sequence of similarities representing similar parts of sentences and differences representing differing parts. The structure of a match sequence that is extracted from a translation example pair is as the following:

$S^1_0D^1_0S^1_1 \ldots D^1_{n-1}S^1_n \leftrightarrow S^2_0D^2_0S^2_1 \ldots D^2_{m-1}S^2_m$ where n,m>1

Here S stands for similarities and D stands for differences. The left side of the arrow is the match sequence for source language sentences and the right is the match sequence for target language sentences.

For example, assume that following two examples E_1 and E_2 are given.

E_1: ыстық шай ішемін ↔ sıcak çay içeceğim (I will drink hot tea)

E_2: ыстық су ішемін ↔ sıcak su içeceğim (I will drink hot water)

As observed in these two pairs, both sides have two similar parts and one differing part. Similar parts in Kazakh side are **ыстық** (hot) and **ішемін** (I will drink), and similar parts in Turkish side are **sıcak** (hot) and **içeceğim** (I will drink). The differing parts are **(шай, су)** in Kazakh side and **(çay, su)** in Turkish side, and they mean **tea** and **water** in English. The match sequence for these examples is obtained as follows.

ыстық (шай, су) ішемін ↔ sıcak (çay, su) içeceğim

Since there is only one difference on both sides, surely these differing parts must be translations of each other. By generalizing this match sequence and by replacing differing parts with variables, the following general translation template and two atomic translation templates from the correspondences of differing parts are inferred.

ыстық X^1 ішемін ↔ sıcak Y^1 içeceğim **if** $X^1 \leftrightarrow Y^1$

шай ↔ çay (tea)

су ↔ su (water)

In general translation templates, the variables that replace differences in Kazakh language side are denoted by X, and Turkish part variables are denoted by Y. The correspondence of two variables is indicated by same superscripted numbers. Although this example is given in surface level, the actual templates in the system are learned at lexical level.

In order to learn translation templates from a match sequence, it has to have exactly n differing parts at both sides and the correspondences of n-1 differing

parts must be determined from previously learned templates. For better comprehension, a formal description of the Translation Template Learning (TTL) algorithm is summarized in Figure 12 as given in [63].

procedure TTL($M_{a,b}$)
begin

- Let the match sequence $M_{a,b}$ be:
  $S_0^1 D_0^1, \ldots, D_{n-1}^1 S_n^1 \leftrightarrow D_0^2 S_0^2, \ldots, D_{m-1}^2 S_m^2$
  if n=m=1 **then**
  - return the following rules
    $S_0^1\ X^1\ S_1^1 \leftrightarrow S_0^2\ X^2\ S_1^2$
    if $X^1 \leftrightarrow X^2$
    $D_{0,a}^1 \leftrightarrow D_{0,a}^2$
    $D_{0,b}^1 \leftrightarrow D_{0,b}^2$

else if 1<n=m **and** n-1 corresponding differences can be found in $M_{a,b}$ **then**

- Assume that the unchecked corresponding differences is $((D_{k_n,a}^1, D_{k_n,b}^1), (D_{l_n,a}^2, D_{l_n,b}^2))$
- Let the list of corresponding differences be $(D_{k_1}^1, D_{l_1}^2) \ldots (D_{k_n}^1, D_{l_n}^2)$ including unchecked ones.
- For each corresponding difference $(D_{k_i}^1, D_{l_i}^2)$ replace $D_{k_i}^1$ with X_i^1 and $D_{k_i}^2$ with X_i^2 to get the new match sequence $M_{a,b}DV$.
- return the following rules:
  $M_{a,b}DV$
  if $X_l^1 \leftrightarrow X_l^2$ and ... and $X_n^1 \leftrightarrow X_n^2$
  $D_{k_n,a}^1 \leftrightarrow D_{l_n,a}^2$
  $D_{k_n,b}^1 \leftrightarrow D_{l_n,b}^2$

end

Figure 15. The TTL Algorithm

Let us look at the following examples at surface level:

E_3: ыстық шай ішеміз ↔ sıcak çay içeceğiz (We will drink hot tea)

E_4: жылы су ішемін ↔ ılık su içeceğim (I will drink warm water)

Although these examples have similar structures, their similarity cannot be observed at their surface level representations. Let us look at the following examples at lexical level after they are morphologically analyzed and disambiguated.

E_3: ыстық+Adj шай+Noun+A3sg+Pnon+Nom іш+Verb+Pos+FutTr+A1pl

↔ sıcak+Adj çay+Noun+A3sg+Pnon+Nom iç+Verb+Pos+Fut+A1pl

E_4: жылы+Adj су+Noun+A3sg+Pnon+Nom іш+Verb+Pos+FutTr+A1sg

↔ ılık+Adj su+Noun+A3sg+Pnon+Nom iç+Verb+Pos+Fut+A1sg

The match sequence which is extracted from E_3 and E_4 is as follows:

(ыстық,жылы)+Adj(шай,су)+Noun+A3sg+Pnon+Nom
іш+Verb+Pos+FutTr(+A1pl,+A1sg)

↔ (sıcak,ılık)+Adj (çay, su)+Noun+A3sg+Pnon+Nom iç+Verb+Pos+Fut(+A1pl, +A1sg)

In this match sequence, there are 3 differences in both sides. Assume that we previously learned following templates.

шай ↔ çay (tea)

су ↔ su (water)

+A1pl ↔ +A1pl (first person plural agreement)

+A1sg ↔ +A1sg (first person singular agreement)

These previously learned templates indicate that the correspondence between the differences **(шай, су)** and **(çay, su)**, and the correspondence between the differences **(+A1pl,+A1sg)** and **(+A1pl,+A1sg)** are known. The last differences **(ыстық, жылы)** and **(sıcak, ılık)** must correspond to each other. In this case the following three templates can be induced from this match sequence.

X^1+Adj X^2+Noun+A3sg+Pnon+Nom іш+Verb+Pos+FutTr X^3

↔ Y^1+Adj Y^2+Noun+A3sg+Pnon+Nom iç+Verb+Pos+Fut Y^3 **if** $X^1 \leftrightarrow Y^1$ **and** $X^2 \leftrightarrow Y^2$ **and** $X^3 \leftrightarrow Y^3$

ыстық ↔ sıcak (hot)

жылы ↔ ılık (warm)

The TTL algorithm described above in this section is applicable to the match sequences, if the number of differences on both sides are equal. If a match sequence will have different number of differences then before applying TTL algorithm, the number of differences should be equated by separating

differences. Assume that the match sequence has less number of differences in its one side than the number of differences in the other side. Then, first of all, constituents at the side with less differences are examined if they contain more than one morpheme. If it is, then the number of differences should be increased by dividing morphemes into smaller parts from morpheme boundaries. After equating the numbers of differences in both sides the TTL algorithm is applied to each pair of examples in the training set to infer translation templates. The algorithm is applied in each stage or turn of learning phase until no new translation template is learned. Thus, after one turn if the number of new learned translation templates does not increase then the learning process stops. If the number of examples is n, then the maximum number of turns of the learning phase is theoretically n-1. However, in practic usually the number of turns needed is much less.

5.1. Type Associated Translation Templates

Variables in general translation templates can be replaced with any string during the translation process. Some of these replacements cannot be appropriate in certain contexts because the variables in learned templates are generalizations of the tokens (stem words or morphemes) in differences and these variables should be replaced with the strings whose types are compatible with the tokens in differences. For this reason, translation templates with typed variables are also learned in our EBMT system same as EBMT system between Turkish and English [63]. Let us consider the following translation examples,

E_5: келдім ↔ geldim (I came)

E_6: кеттім ↔ gittim (I went)

and their lexical level representations.

E_5: кел+Verb+Pos+PastDF+A1sg ↔ gel+Verb+Pos+Past+A1sg

E_6: кет+Verb+Pos+PastDF+A1sg ↔ git+Verb+Pos+Past+A1sg

From the match sequence of above examples, the following general translation template is learned by replacing differences (кел, кет) and (gel, git) (*come* and *go* in English) with appropriate variables.

X^1+Verb+Pos+PastDF+A1sg ↔ Y^1+Verb+Pos+Past+A1sg if $X^1 \leftrightarrow Y^1$

Now, let us assume that we have previously learned the following atomic translation template

той ↔ düğün (wedding)

where both Kazakh and Turkish words are nouns. When we fed our system with the lexical form of the sentence **тойдым** (I am full up) whose lexical form is **той+Verb+Pos+PastDF+A1sg** in order to translate it to Turkish. Using the above templates and replacing X^1 with **той** and Y^1 with **düğün**, the invalid translation result **düğün +Verb+Pos+Past+A1sg** is obtained since **düğün** stem is a Turkish noun not a verb. In order to not forget types of strings in differences that are replaced with variables during learning, we also infer general translation templates with typed variables. The type of a variable is obtained by finding the youngest common ancestor of the tokens in the difference that is replaced by the variable. The youngest common ancestor of two tokens is the first common ancestor in the morphology type lattice of the language of those tokens.

5.2. Lattice

Lattice means an arrangement of particles in a regular periodic and cross-cutting interlaced pattern. Here linguistic part-of-speech tag arrangement model is defined as a lattice. It is required and suitable for hierarchical characteristic of part-of-speech tags and there can be some cross-cutting types that belongs to more than one category. A common category for two different types can be found by constructing a lattice like structure which resembles an undirected acyclic graph. There are two lattice structure are used for both languages. Figure 6, shows the structure of the lattice that we have used in our system for Kazakh language which was used earlier for Turkish language [63]. The main lattice structures used for Kazakh and Turkish are given in Appendices 4 and 5 respectively. Both languages are agglutinative type and have similar lattice structure with some difference according to categorizations which are taken from the morphological analyzers.

The main categorization depends on the affixes. There are main categories such as noun, verb etc. and they have supertype parent ANY. However, affixes are grouped according to the categories that an affix can follow. In both languages every noun can appear as adjective and every adjective as noun,

NOUN-SUFFIX category is under ADJ-SUFFIX category also. An other categorization principle for both languages is the categorization of the affixes causing derivation. Derivative words are frequently seen type of words in Turkish and in Kazakh. Derivation can be defined as creating new morphemes from other morphemes by adding a derivational affix. So this kind of affixes are grouped according to the main categories that they can follow and the category of the word that they yield after the derivation process. For example the morpheme ”сыз” derives a noun “ұят” (shame) to an adjective “ұятсыз” (barefaced) and its lexical equivalent is ˆDB+Adj+Without. So, the category ˆDB+Adj+Without has parent NOUN-DB-ADJ.

Using a tree structure is not suitable for indicating multiple parents for some constituents such as the case of “+A3sg” which can both appear as the singular noun agreement and the third person singular verb agreement. Here the leafs given in circle are the constituents whose type is the first parent indicated by rectangle of that leaf.

If a morpheme is not a root then its type label would be its own name. On the other case, its type label will be type of the root word. For example, “book+Noun+A3sg+Pnon+Nom”, here “book+Noun” would have simply a type label “Noun” and other two morphemes will have type labels as their names, “A3sg” and “Nom” respectively.

A label to token pairs is assigned by finding the nearest common parent of the two tokens. For example, the difference D_i: (кел+Verb, кет+Verb) to be replaced by a variable by learning algorithm. Since, there are single tokens in the difference and their types match the variable with type label would be X_{Verb} . However it is the simplest case and in situations where pair tokens do not match in terms of type, learning algorithm searches for nearest supertype of pairs. Then, the concatenation of each such token pair in the difference becomes the type label of a variable. As an example of a difference, assume that D_i is given and a part of it in Kazakh Lattice is illustrated in Figure 13:

D_i: (той+Noun+A3sg, мен+Pron+A1sg)

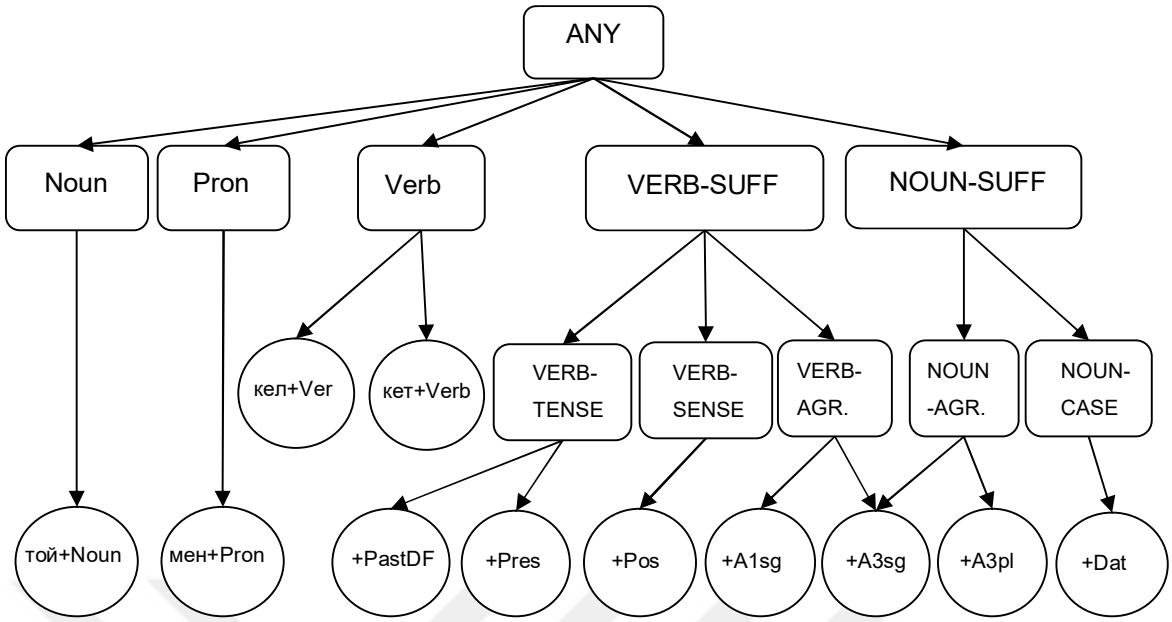


Figure 16. A Part of a Kazakh Lattice

Here, the first token pair is (той+Noun, мен+Pron) and their first common parent type is “ANY”. Likewise, the type label of the second token pair (+A3sg, +A1sg) will be VERB-AGREEMENT. So the difference Di would have label of the variable as “ANY VERB-AGREEMENT” which is the concatenation of types of each token pair.

Mentioning about our previous example, the general translation template above can be learned as the following type associated translation template.

X^1_{Verb} +Verb+Pos+PastDF+A1sg ↔Y^1_{Verb}+Verb+Pos+Past+A1sg

if $X^1_{Verb} \leftrightarrow Y^1_{Verb}$

The variable X^1_{Verb} can only be replaceable with a Kazakh verb and the variable Y^1_{Verb} can only be replaceable with a Turkish verb. Since the variable X^1 replaces the difference (кел, кет), and the youngest common ancestor of the tokens of this difference is **Verb** node in the Kazakh type lattice. Similarly, the tokens of the difference (gel,git) that is replaced by the variable Y^1 is **Verb** node in the Turkish type lattice.

5.3. Epsilon Insertion

Thus, the type of a variable is infered by pair-wise searching a type in the lattice. It is not a troublesome to find a type label for a variable if pairs in difference have same number of tokens. The problem is how to decide the pairs when there are different number of constituents. Epsilon is inserted to the side with fewer number of tokens. The insertion point of an epsilon can be determined by calculating a generalization score for each of the possible insertion points and then choosing the one with the least score. First, epsilon is added to every possible point. Taking into account that the distance from epsilon to any type is set to 2, the distances between tokens are calculated using lattice structure. It is the length of the shortest path between types. Then, the sum of these distances between the types of the corresponding tokens in the constituents is called the generalization score of an epsilon insertion point possibility. For example, given following match sequence and the learning algorithm has to assign a type label to the variable:

E_1:(көк+Adj, сары+Adj) машина+Noun +A3sg +Pnon +Nom

↔ (bir+Num+Card mavi+Adj, sarı+Adj) araba+Noun +A3sg +Pnon +Nom

The difference given in E_1 has single token on the left-hand side and there are 2 tokens in both of the constituents. Thus, epsilon insertion is unnecessary here. However, 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.

For example,

(bir+Num+Card mavi+Adj, ε sarı+Adj)

(bir+Num+Card mavi+Adj, sarı+Adj ε)

In order to calculate best epsilon position the lattice given in Figure 14 is used. For every possibility, we calculate the shortest distance of the types pairwise and chose the possibility with the minimum value.

Thus, the generalization scores for the two epsilon position points are calculated as:

genScore1 = minDist(bir+Num+Card, ε) + minDist(mavi+Adj, sarı+Adj)= 2+2=4

genScore2 = minDist(bir+Num+Card, sarı+Adj) + minDist(mavi+Adj, ε)= 4+2=6

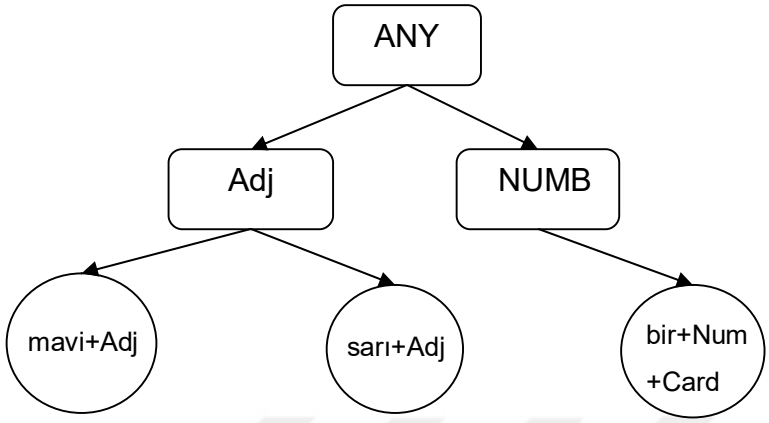


Figure 17. Part of a Turkish Lattice

taken as As it is seen, the most appropriate possibility is the first one since it has the smallest generalization score. So the induced translation template for the match sequence E_1, will be shown as T_1 and T_2

E_1:(көк+Adj, сары+Adj) машина+Noun +A3sg +Pnon +Nom

↔ (bir+Num+Card mavi+Adj, sarı+Adj) araba+Noun +A3sg +Pnon +Nom

E_1:X^1_{Adj} машина+Noun +A3sg +Pnon +Nom ↔

$Y^1_{nullor(NumCard)\ Adj}$ kitap+Noun +A3sg +Pnon +Nom

$T_{1:}$ көк+Adj ↔ bir+Num+Card mavi+Adj

T_2:сары+Adj ↔ sarı+Adj

Here, the nullor defines the token position as nullable, that is either epsilon or a cardinal number can be substitude to the token position during the translation phase.

5.4. Learning From Learned Templates

In the previous sections, it is presented that effective learning is obtained by extracting translation templates from translation example pairs. Size of learned templates are usually limited because of scarceness of prealigned example pairs. In order to increase the geenrality of learned templates and learning effectiveness previously learned templates can be used for learning purpose. Actually this is the generalization of two similar translation templates into one template. Then two sentences which are generalized to form the last learned and more general translation template will be able to be represented by their general form.

For example, assume that we have translation examples given below. The first thing to do is to extract a match sequence from these examples.

E_1:бала+Noun+A3pl+Pnon+Nom кел+Verb+Pos+PastDF+A3sg

↔oğlan+Noun+A3pl+Pnon+Nom gel+Verb+Pos+Past+A3pl

E_2:бала+Noun+A3pl+Pnon+Nom кет+Verb+Neg+PastDF+A3sg

↔oğlan+Noun+A3pl+Pnon+Nom git+Verb+Neg+Past+A3pl

First of all from above examples we will have match sequence M1 and then template as T1.

M_1:бала+Noun+A3pl+Pnon+Nom(кел+Verb+Pos,
кет+Verb+Neg)+PastDF+A3sg

↔oğlan+Noun+A3pl+Pnon+Nom (gel+Verb+Pos, git+Verb+Neg)+Past+A3pl

T1: бала+Noun+A3pl+Pnon+Nom (X^1 $_{\text{Verb VERB-SENSE}}$)+PastDF+A3sg

↔oğlan+Noun+A3pl+Pnon+Nom (Y^1 $_{\text{Verb VERB-SENSE}}$)+Past+A3pl

Also, assume that we have other two examples such as E_3 and E_4 and template T_2 extracted from these examples :

E_3:қыз+Noun+A3pl+Pnon+Nom кел+Verb+Pos+PastDF+A3sg

↔kız+Noun+A3pl+Pnon+Nom gel+Verb+Pos+Past+A3pl

E_4:қыз+Noun+A3pl+Pnon+Nom кет+Verb+Neg+PastDF+A3sg

↔kız+Noun+A3pl+Pnon+Nom git+Verb+Neg+Past+A3pl

T_2: қыз+Noun+A3pl+Pnon+Nom (X^1 $_{Verb\ VERB\text{-}SENSE}$)+PastDF+A3sg

↔kız+Noun+A3pl+Pnon+Nom (Y^1 $_{Verb\ VERB\text{-}SENSE}$)+Past+A3pl

As mentioning before, T1 and T2 are templates learned before and system will learn again from these templates to improve learning ability. Thus, T_G is the generalized learned template given in 5.40.

T_G: (X^1_{Noun})+A3pl+Pnon+Nom (X^2 $_{Verb\ VERB\text{-}SENSE}$)+PastDF+A3sg

↔(Y^1_{Noun})+A3pl+Pnon+Nom (Y^2 $_{Verb\ VERB\text{-}SENSE}$)+Past+A3pl (5.40)

Therefore, templates T1 and T2 are merged to form TG. After giving all details of learning algorithm it can be summarized by flowchart given in Figure 15

5.5. Confidence Factors

A learned translation template may not produce a correct translation at all cases. In some of its applicable cases, it may generate invalid results. We want to know the confidence level of a translation template and its confidence level is estimated using the statistical information available in translation examples in the training set. The confidence levels of a template can be different in two translation directions. For this reason, we associate each learned translation template with two confidence factors (CF_{12},CF_{21}) where CF_{12} is a confidence factor to be used in translations from L1 to L2 and CF_{21} is to be used in translations from L2 to L1. We have implemented a type associated version of calculating confidence factors as worked in an earlier system [37].

The confidence factors of templates are calculated using their matching performances with translation examples in the training set. We say that the language-1 side of a translation template matches with a translation example if it matches with a substring of that example in the same language side. A side of an atomic translation template matches with a translation example side if it is a substring of that example side. On the other hand, a side of a general template with typed variables can only match with a translation example side if it matches with that example side bounding its variables to strings of that example and those strings satisfy types of those variables.

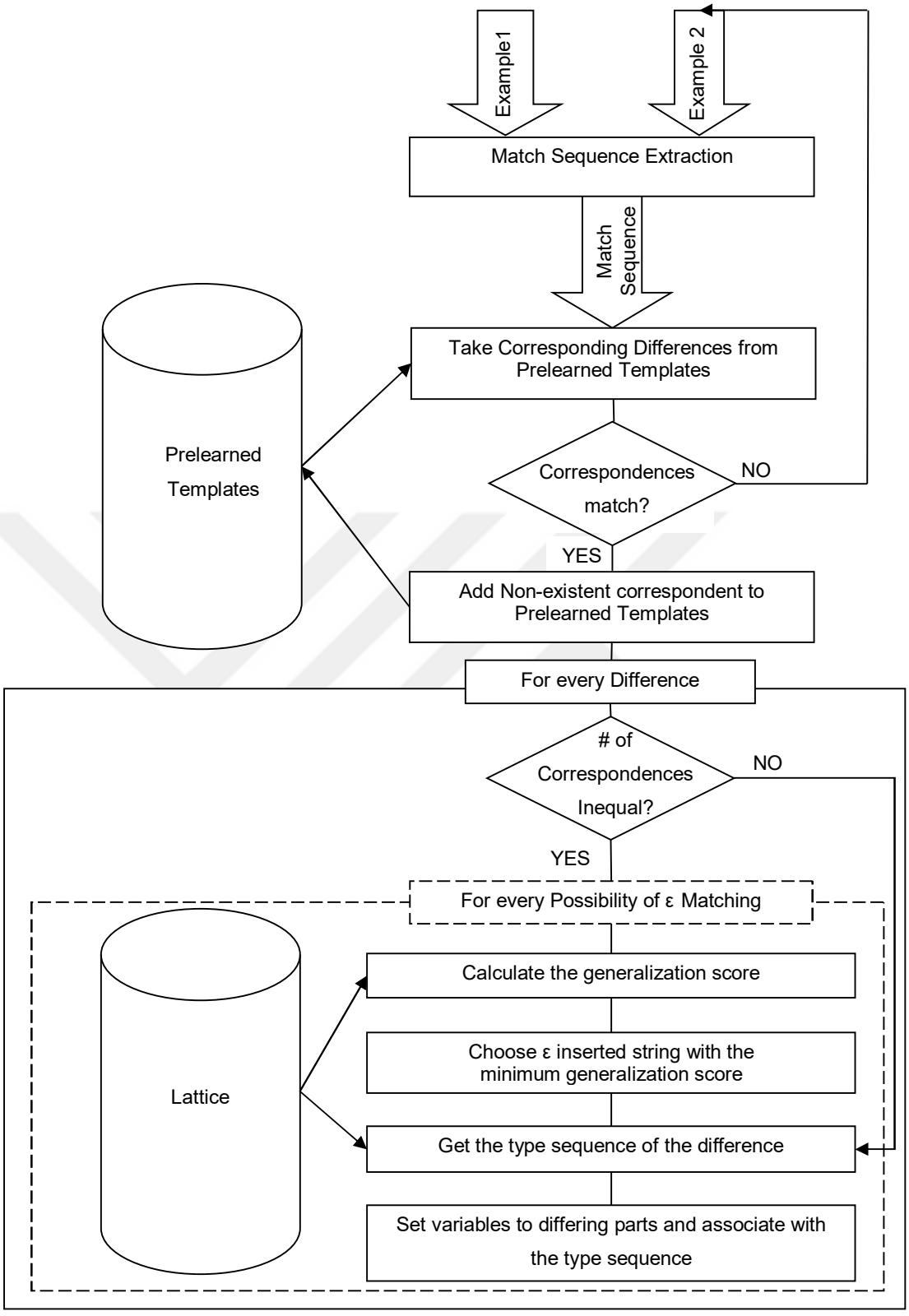


Figure 18. Flowchart of Learning Algorithm

Let us assume that we have a translation template which has “L1Side ↔ L2Side” form, its confidence factors (CF_{12},CF_{21}) are computed as follows:

$CF_{12} = N_{12} / (N_{12} + N_1)$

$CF_{21} = N_{12} / (N_{12} + N_2)$

where N_{12} is the number of translation examples such that they match with both sides of the translation template, N_1 is the number of translation examples such that they match with L1Side but they do not match with L2Side, and N_2 is the number of translation examples such that they match with L2Side but they do not match with L1Side. For clarification, assume that confidence factor of template such as :“маған X^1_{Verb} ↔bana Y^1_{Verb}” is to be calculated from following training examples.

E_1: оған қара+Verb+Pos+ImMD+A2Sg ↔ona bak*(look at him)*

E_2: ол қара+Adj бала ↔o esmer çocuk *(he is brunette)*

E_3: сен қара+Verb+Pos+Past+A2Sg ↔ sen baktın *(you looked)*

E_4: маған қара+Verb+Pos^DB+Adj кісі ↔bana bakan kişi *(person who looked at me)*

Thus, from above examples we have calculated confidence factor of type associated templates as:

E_1 matches translation template for both languages, so N1 is increased by 1

E_2 doesn't match translation template

E_3 matches translation template for both sides, so N1 is increased by 1

E_4 doesn't match translation template

Thus, confidence factor from Kazakh to Turkish is calculated as:

$confidence\,factor^1 = \frac{2}{2+0} = 1.0$ and from Turkish to Kazakh

$confidence\,factor^2 = \frac{2}{2+0} = 1.0$

6. TRANSLATION WITH LEARNED TRANSLATION TEMPLATES

Learned translation templates are bi-directional and they can be used in translations of new sentences in both directions. This means that a Kazakh sentence can be translated into Turkish and a Turkish sentence can be translated into Kazakh language using learned templates. Left sides of learned templates define a grammar of Kazakh language and right sides define a Turkish grammar. The translation process can be thought as a grammar correctness check of a given source language sentence by the defined source language grammar through learned templates and at the same time generating its translation in target language. First, the given source language sentence is morphologically analyzed and disambiguated, and the translation process is performed by an Earley parser [81] which parses the source language sentence at lexical level and generates its translation at lexical level in target language. After translation results are sorted using confidence factors and a SLM of the target language, the surface level of the top translation result at lexical level is produced using the target language morphological generator.

If the given source language segment is translated using an atomic translation template, its translation is the target language part of that template. In order to translate a source language segment with a general translation template, the given segment must match with the source language part of that general template by binding source language variables to corresponding parts in the given source language segment. Bound strings must satisfy type constraints of those variables, and the matching process is unsuccessful if they do not. Strings bound to source language variables are recursively translated, and their results are bound to corresponding target language variables. In order to have a successful translation process with this general template, sub-translation results that are bound to target language variables must also satisfy type constraints of those variables.

A confidence factor value for a translation result is calculated using the confidence factors of the translation templates involved in that translation. As mentioned before, each translation template is attached with to two confidence factors (CF_{12},CF_{21}) where CF_{12} is a confidence factor to be used in translations from L1 to L2 and CF_{21} is to be used in translations from L2 to L1. In the

calculation of confidence factors of translation results, confidence factors of templates in translation direction are used. If a translation result is directly obtained with an atomic translation template, the confidence value of that result is same as the confidence factor of that atomic template. If a translation result is obtained using a general template, the confidence value of the translation result is the multiplication of the confidence factor of that template together with confidence values of sub-translation results which are obtained from translation of strings bound to source language variables.

6.1. Matching Phase

The translation process is mainly a matching process which aims to find best match in terms of parses at lexical level with type constraints and the available example in the database. The basic idea is to construct bilingual aligned corpus and using a suitable algorithm for matching process. In thesis, the translation process is performed by an Earley parser [81] which parses the source language sentence at lexical level and generates its translation at lexical level in target language.

6.2. Earley Parser

The system, after being trained with parallel alighned corpora and learned translation templates, it is able to match in order to find an appropriate templates for a given sentence to be translated. This is performed by known dynamic algorithm which is called Earley Parsing algorithm. As known, Earley Parsing algorithm uses chart to eliminate backtracking and composed of three main components such as: Predictor, Scanner and Completor. Also, parser performs matching of a given sentence according to predefined grammar rules with dot notation method. In our system rules are prelearned templates with type information of variables and confidence factors. For example, lets look at parsing example which is used in our system given in Table 10.

If we have sentence to parse “қыздар келеді →kızlar gelecekler” which means *girls will come* in surface form. Giving lexical form such as: **қыз+Noun +A3pl +Pnon +Nom кел+Verb +Pos +FutTR +A3pl**

Table 10. Illustration of Parsing

Rule	Template	Types	CF
S→S +A3pl +Pnon +Nom S +Pos +FutTR +A3pl	[S +A3pl +Pnon +Nom S +Fut +A3pl]	(Noun ,Noun) (Verb ,Verb)	[1.0,1.0]
S →қыз+Noun	[kız+Noun]		[1.0,1.0]
S → кел+Verb	[gel+Verb]		[1.0,1.0]

Here every tag will be considered as seperate word while parsing and will have 8 charts as following:

Table 11. Earley Parser Example

Chart #	Rule #	Dotted Rule	Start-End	Parents
Chart 0	1	S → . S	[0-0]	
	2	S → . қыз+Noun	[0-0]	
Chart 1	3	S → қыз+Noun .	[0-1]	
	4	S→ S .+A3pl +Pnon +Nom S +Pos +FutTR +A3pl	[0-1]	[3]
Chart 2	5	S → S +A3pl .+Pnon +Nom S +Pos +FutTR +A3pl	[0-2]	[3]
Chart 3	6	S → S +A3pl +Pnon .+Nom S +Pos +FutTR +A3pl	[0-3]	[3]
Chart 4	7	S → S +A3pl +Pnon +Nom .S +Pos +FutTR +A3pl	[0-4]	[3]
	8	S → .кел+Verb	[4-4]	
	9	S → . S	[4-4]	
Chart 5	10	S → кел+Verb .	[4-5]	
	11	S → S +A3pl +Pnon +Nom S .+Pos +FutTR +A3pl	[0-5]	[3,10]
	12	S → S .+A3pl +Pnon +Nom S +Pos +FutTR +A3pl	[4-5]	[10]
Chart 6	13	S → S +A3pl +Pnon +Nom S +Pos .+FutTR +A3pl	[0-6]	[3,10]
Chart 7	14	S → S +A3pl +Pnon +Nom S +Pos +FutTR .+A3pl	[0-7]	[3,10]
Chart 8	15	S → S +A3pl +Pnon +Nom S +Pos +FutTR +A3pl .	[0-8]	[3,10]
	16	S → S .+A3pl +Pnon +Nom S +Pos +FutTR +A3pl	[0-8]	[15]

Result is as “1 Translation found from rule 15 : kız+Noun +A3pl +Pnon +Nom gel+Verb +Pos +Fut +A3pl”

Parse tree is obtained by parents of rule. Here from rule 15 they are 3 and 10.

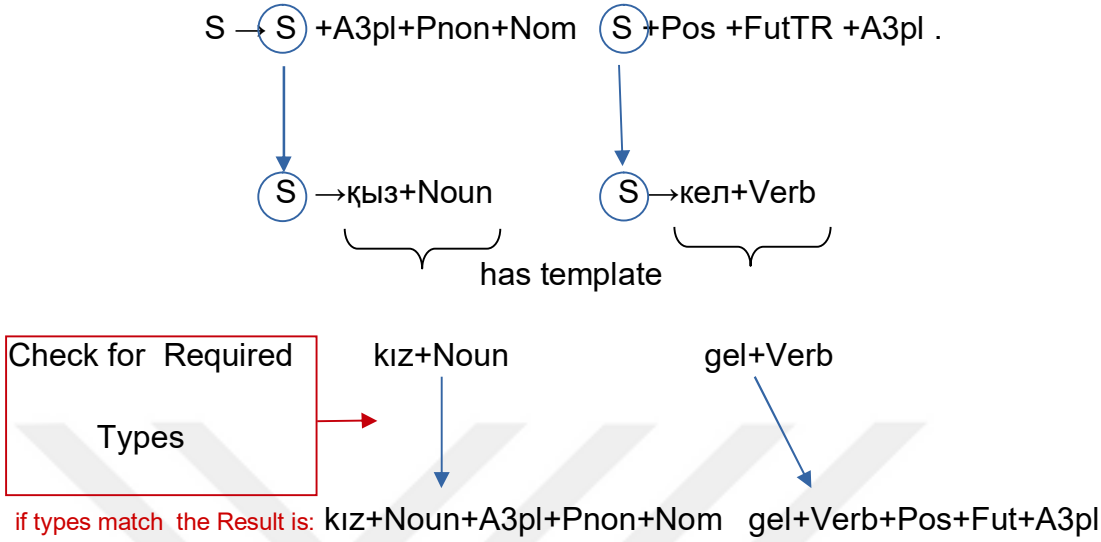


Figure 19. Parse tree of Earley Algorithm

7. STATISTICAL LANGUAGE MODEL SUPPORTED TRANSLATION ORDERING

Translations are performed using the translation templates which are learned from translation examples in a training set. Although the size of training set can be big enough to learn many useful translation templates so that these learned templates can be capable to translate many new sentences, it cannot be big enough to capture all variations in languages. In the EBMT part of our system, translation results can be ordered according to their confidence values that are computed using confidence factors of learned templates. Confidence factors are calculated according to the statistical information available in bi-lingual translation examples of the training set and they may not catch some usage of text in target language because of its limited size.

In order to capture more statistical information about languages and use this information in the orderings of translation results, we suggest the usage of bigger monolingual corpus to collect this statistical knowledge as Statistical Language Models (SLMs). The size of monolingual corpora is much bigger than the size of bilingual corpus of translation examples. Translation results are ordered using their confidence values and their target language usage probabilities which are calculated using a SLM. Thus, the combination of the available knowledge in both the bilingual corpus and the monolingual corpus of the target language plays a role in the orderings of translation results in our hybrid EBMT system.

Turkish and Kazakh languages are closely related languages, but in some case they have considerably many semantical differences. For example, two of possible Turkish translations of Kazakh verb **тозды** are **eskidi** (worn out) and **yıprandı** (exhausted), and one of them can be more appropriate translation depending on the subject of the sentence. Since its most possible translation is **eskidi**, translation templates doing this translation can have bigger confidence factors than confidence factors of translation templates translating that verb as **yıprandı**. Then, whenever we want to translate **жүйкем тозды** (I had nervous exhaustion) to Turkish language, it will be translated as “sinirim eskidi” with higher confidence value and it is not a valid translation result in Turkish. Here, the more correct Turkish translation is **sinirim yıprandı** which will have a less

confidence value. According to the usage probabilities of these two translation results which can be calculated using a target language (Turkish) SLM, the probability of the translation result **sinirim yıprandı** can be higher. Thus, a common usage of text can be caught by possessing a high sentence probability.

SLMs can be constructed using surface level forms of words or their lexical level forms. Since both of our languages are agglutinative languages with complex morphology structures, SLMs depending on lexical level forms are more appropriate. Since agglutinative languages are suffering from data sparseness, it is stated that better performance is observed while a language model is constructed according to morphological information than using in surface form [53].

7.1. Language Models

We created three language models to be taken into account when calculating probabilities of sentences. Language Model 1 (LM1) takes stems of words together with their final tags as tokens in the model creation, Language Model 2 (LM2) takes stems of words together with their main part of speech and Language Model 3 (LM3) takes word's final tag without a stem.

For instance, let us consider Turkish sentence **evim eskidi** (my house became old). All sentence words are morphologically analyzed and disambiguated to obtain words at lexical level as follows:

ev +Noun+A3sg+P1sg+Nom eski +Verb+Pos+Past+A3sg

Then, each model creates own tokens from these lexical level representations as follows:

LM1: <s> ev +Noun+A3sg+P1sg+Nom eski +Verb+Pos+Past+A3sg </s>

LM2: <s> ev +Noun eski +Verb </s>

LM3: <s> Noun+A3sg+P1sg+Nom Verb+Pos+Past+A3sg </s>

Here <s> and </s> are sentence boundaries to calculate the probability of a sentence.

7.2. Sentence Probability

From the monolingual corpus of each language, bigram and unigram probabilities of all language models are calculated using AntConc software [67]. A SLM is a probability distribution P(s) over text that gives how frequently given n-grams occur and we calculate bigram probabilities. We use standard bigram formula in bigram calculations. Thus, we obtain three language models for each language.

$$P(w_i | w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

For example, let's illustrate bigram calculation of a sentence. Assume that a sentence produced as a translation result from a system is **<s>мен үйге келдім</s>** which means "*I came home".* If to calculate bigram probability of the sentence according to language model of Type 2(LM2). In order to present a calculation of bigram of the sentence, we use a chart. Due to its language model type we have to take types of stem as their tag. It is **<s>мен+Pron үй+Noun кел+Verb</s>**. Also, we have example text for calculating bigrams such as:

<s>мен+Pron мектеп+Noun кел+Verb</s>

<s>сен+Pron үй+Noun кел+Verb</s>

<s>ол+Pron бақша+Noun бар+Verb</s>

<s>мен+Pron жұмыс+Noun кет+Verb</s>

<s>мен+Pron үй+Noun кет+Verb</s> (7.20)

Table 12. Calculation of sentence probability

	<s>	мен+Pron	үй+Noun	кел+Verb	</s>
Bigrams		3	1	1	2
Unigrams	5	3	2	2	5
Prob. Of Sentence	0.09 =	3/5 *	1/3 *	1/2 *	2/2

So, the probability of sentence is 0.09 calculated according to LM2 and examples in 7.20 .

In translation process, we also calculate usage probabilities of translation results using a selected language model of the target language. Whenever we have more than one translation result, we look at their confidence values and usage probabilities in order to decide the best one. Since training examples are limited than any other collected monolingual data in both languages, we use sentence probabilities obtained from their SLMs as a second reference. We calculate a combined score for each translation result in order to sort translation results with respect to their combined scores obtained from confidence values and usage probabilities that are weighted with two coefficients α, β. Thus, a combined score of a translation result with a confidence value CF and a usage probability LM is equal to α*CF+β*LM, which will be explained in Chapter 8.

7.3. Smoothing

Language models assign a probability that a sentence is a legal string in a language. Since sparseness of agglutinative language data is the main problem which we are faced with, there can be combinations of words that never occur in training. If a new combination occurs during translation, it is given a probability of zero and the entire sequence gets a probability of zero. Therefore some smoothing techniques should be used to prevent zero probability. In this work we used “Add--1” or Laplace smoothing technique which is the easiest one among other techniques. Thus, pretend that there is one more word than it is in real and just add one to all the counts as can be seen from formula given below:

$$P_{Add-1}(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i) + 1}{C(w_{i-1}) + U}$$

For example, assume that there are words such as *a, b, c* and *d*. Also, four kinds of occurrences exist as following:

#(aa)=2 → Probability of Sentence= 2/6

#(ab)=1 → Probability of Sentence= 1/6

#(ac)=0 → Probability of Sentence= 0/6

#(ad)=3 → Probability of Sentence= 3/6

In order to smooth or regularize probabilities of unseen word, we add 1 to the nominator and add number of unique words to the denominator. Thus, according to above example we will have:

(2+1)/(6+4)=3/10

(1+1)/(6+4)=2/10

(0+1)/(6+4)=1/10

(3+1)/(6+4)=4/10

Therefore, we have prevent zero probability, instead small amount of count was added by proportionally changing other probabilities.

8. EVALUATIONS AND ANALYSIS OF EVALUATION RESULTS

8.1. α and β Ranking

We calculate a combined score for each translation result in order to sort translation results with respect to their combined scores obtained from confidence values and usage probabilities that are weighted with two coefficients α, β. Thus, a combined score of a translation result with a confidence value CF and a usage probability LM is equal to α*CF+β*LM.

We want to find best matching coefficients α, β for each language model so that we can get best performance from our hybrid EBMT system. If α is 1 and β is 0, the decision is only made by the EBMT part of the system. On the other hand, the decision is only made by SLM if α is 0 and β is 1. Of course, the best combination is the combination that produces best results. In order to choose best coefficient combination for coefficients, we look at BLUE score [82] of the system on a test set of bilingual translation examples.

Table 13. Example for Calculation Coefficients α and β

Result	Conf. Factor	Lang. Model 1	Lang. Model 2	Lang. Model 3	BLEU
R_1	0.08	0.05	0.05	0.06	1
R_2	0.04	0.07	0.08	0.04	0.61
…					
R_n	0.01	0.03	0.04	0.03	0.74

8.2. BLEU Method

BLEU score is a reference value (from 0 to 1) which identifies the closeness of a result generated by machine translation to human translation independent from language information, and it is a n-gram precision comparison method that compares a translation result with human reference translations. If machine translation is closer to human translation then score is higher. The quality of translation is a number between 0 and 1, and this measurement is a statistical closeness to a set of perfect translations. Computing the n-gram value (taking n as 1) can give an absurd results if some modifications are not made. Thus, first the maximum number of word which occurs in any single reference translation

is counted. Then, for each candidate word the total count is clipped by its maximum reference. At last, these clipped counts are added and divided by total number of candidate words. In summary, BLEU score calculations performed by taking parameter n as 1,...,4 and using the geometric mean of these precisions as following: [82].

$$p_n = \frac{\sum_{C\in\{Candidates\}} \sum_{n-gram\in C} Count_{clip}(n-gram)}{\sum_{C'\in\{Candidates\}} \sum_{n-gram'\in C'} Count(n-gram')}$$

Moreover according to [82], another penalty called Brevity Penalty has been added for the longer candidate sentence than reference in BLEU score calculations and we used it in our calculations too. This is calculated as:

$$BP = \begin{cases} 1 & if\ c > r \\ e^{(1-r/c)} & if\ c \leq r \end{cases}$$

Then, the BLEU score is calculated as shown in Equation 6.3, and the positive weights of n-grams w_n sums up to 1, if n=4 then w_4=0,25.

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} w_n \log p_n) \qquad (6.3)$$

For example, assume that our system has translated a sentence “құжатқа қол қой” which means *sign a document* as “belgeye el at” . Then, for a given reference which is “belgeye imza at” precisions for n=1,2,3,4 are calculated as stated in formula given in 6.1. The BLEU score is calculated in lexical level. Thus, candidate sentence is “belge+Noun +A3sg +Pnon +Dat el+Noun +A3sg +Pnon +Nom at+Verb +Pos +Imp +A2sg”, here every token treated separately, except type tokens of a word.

The reference sentence is “belge+Noun +A3sg +Pnon +Dat imza+Noun +A3sg +Pnon +Nom at+Verb +Pos +Imp +A2sg”. According to BLEU calculation process,

P_1=10/12

P_2=7\11

P_3=4\10

P_4=1\9

Since candidate sentence has same length with reference sentence Brevity Penalty is 1.

Then, weighted precision is equal to 0,39

In order to determine the best coefficient combination for each language model, two test sets are used. Different combinations of coefficients are tried to find the best combination that produces the best BLUE score for that test set. Results of this evaluation are given in Section 8.4. For each language model, we also give MRR (Mean Reciprocal Rank) values [83] for translation results produced by type associated translation templates.

8.3. Mean Reciprocal Rank

An evaluation of any process that produces a list of possible responses to a sample of queries is performed by statistic measure which is called mean reciprocal rank ordered by probability of correctness. The multiplicative inverse of the rank of the first correct answer is calculated and it is the reciprocal rank of a query response. Then, the average of the reciprocal ranks of results for a sample of queries Q is the mean reciprocal rank, which is illustrated below:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$$

where $rank_i$ refers to the rank position of the first relevant result. For example, assume that our system found 4 translation results for one sentence, such as “құжатқа қол қой” given in Table 14. Also, rank query of results are taken according to BLEU score calculated before. Thus, results are sorted according to their BLEU score and given rank query number due to their positions.

Table 14. Calculation of MRR

Translation results		BLEU	Query	MRR
1	belgeye el at	0.73	2	1/2
2	belgeye imza at	1.0	1	1
3	belgeye el koy	0.39	3	1/3
4	belgeye imza koy	0.73	2	1/2

MRR values reflect the rankings of translation results with highest BLEU scores and a translation result with highest BLEU score is accepted as a true translation result.

8.4. Evaluation

In order to test the performance of our hybrid EBMT system, different training and test sets are composed. A training set that consists of 1000 bilingual aligned translation example pairs is created in order to infer translation templates and their confidence factors from that training set. Another 200 aligned translation example pairs which do not exist in training pairs are used for testing the system by calculating the average BLEU score for the test set. This test contains 100 pairs that are context related with SLM of target language and another 100 pairs that are not context related. The Turkish monolingual corpus contains 300000 words at surface level, and the Kazakh monolingual corpus contains 100000 words. There are 48028 unique tokens in Turkish LM1, 26937 unique tokens in Turkish LM2 and 660 unique tokens in Turkish LM3. There are 20468 unique tokens in Kazakh LM1, 12152 unique tokens in Kazakh LM2 and 361 unique tokens in Kazakh LM3.

We designed two evaluation schemas with our test set in order to observe the results of our hybrid EBMT system. The first evaluation is done with 100 context unrelated translation examples and the second evaluation is done with another 100 context related translation examples. The results of these two evaluations are given in Table 15 and Table 16 for translations from Kazakh language to Turkish. Our MT system works bidirectional and same evaluations are performed for translations from Turkish to Kazakh language. Since they have similar results with the results in Table 15 and Table 16, we do not place them here.

In Table 15 and Table 16, the results are given for each language model for different coefficient constants. For each language model, BLEU scores of the system are

Table 15. Result of context unrelated test set.

α	β	TM Bleu Avg	MRR Avg	TL Bleu Avg	TM Bleu Avg	MRR Avg	TL Bleu Avg	TM Bleu Avg	MRR Avg	TL Bleu Avg
0	1	0.7	0.68	0.64	0.71	0.67	0.64	0.72	0.71	0.69
0.1	0.9	0.7	0.68	0.66	0.71	0.66	0.64	0.72	0.71	0.69
0.2	0.8	0.7	0.68	0.66	0.7	0.68	0.65	0.72	0.71	0.69
0.3	0.7	0.7	0.69	0.66	0.72	0.71	0.65	0.72	0.71	0.67
0.4	0.6	0.72	0.72	0.66	0.72	0.71	0.65	0.72	0.71	0.67
0.5	0.5	0.72	0.72	0.66	0.72	0.71	0.65	0.73	0.74	0.67
0.6	0.4	0.72	0.72	0.66	0.73	0.74	0.65	0.74	0.77	0.67
0.7	0.3	0.73	0.75	0.66	0.74	0.77	0.65	0.74	0.77	0.67
0.8	0.2	0.74	0.77	0.66	0.74	0.77	0.65	0.76	0.79	0.67
0.9	0.1	0.76	0.8	0.65	0.77	0.8	0.65	0.76	0.79	0.67
1	0	0.76	0.8	0.66	0.76	0.79	0.66	0.76	0.79	0.66

given for when translation templates with typed variables are used in translations (TM columns) and when translation templates are used in translations without enforcing any type information imposed by their type variables (TL columns). The reason that we give results in TM and TL columns is to demonstrate that type associated translation templates produce better results than translation templates without type information. This fact can be observed from TM and TL columns of each language model since results in TM columns are consistently higher than results in TL columns.

Table 16. Result of context related test set

α	β	TM Bleu Avg	MRR Avg	TL Bleu Avg	TM Bleu Avg	MRR Avg	TL Bleu Avg	TM Bleu Avg	MRR Avg	TL Bleu Avg
0	1	0.71	0.67	0.63	0.72	0.73	0.61	0.73	0.74	0.67
0.1	0.9	0.72	0.73	0.64	0.72	0.73	0.61	0.74	0.76	0.64
0.2	0.8	0.72	0.73	0.64	0.72	0.73	0.61	0.74	0.76	0.64
0.3	0.7	0.72	0.73	0.64	0.74	0.76	0.62	0.74	0.76	0.64
0.4	0.6	0.74	0.76	0.64	0.74	0.76	0.62	0.74	0.76	0.64
0.5	0.5	0.74	0.76	0.64	0.74	0.76	0.62	0.75	0.8	0.64
0.6	0.4	0.74	0.76	0.64	0.74	0.76	0.62	0.75	0.8	0.64
0.7	0.3	0.75	0.8	0.64	0.75	0.8	0.62	0.75	0.8	0.64
0.8	0.2	0.75	0.8	0.64	0.76	0.81	0.62	0.75	0.8	0.64
0.9	0.1	0.75	0.8	0.64	0.76	0.81	0.62	0.75	0.8	0.64
1	0	0.75	0.8	0.62	0.75	0.8	0.62	0.75	0.8	0.62

For each language model, we also give MRR (Mean Reciprocal Rank) values [83] for translation results produced by type associated translation templates. MRR values reflect the rankings of translation results with highest BLEU scores and a translation result with highest BLEU score is accepted as a true translation result.

Table 17. Enhanced results according to slms of Turkish.

#	**Source Sentence:**құжат+Noun+A3sg+Pnon+Dat қол+Noun+A3sg+Pnon+Nom қой+Verb+Pos+ImpMD+A2sg (құжатқа қол қой-sign a document) **Reference Sentence:** belgeye imza at	CF	LM1	LM2	LM3	BLEU
R1	belge+Noun+A3sg+Pnon+Dat el+Noun+A3sg+Pnon+Nom at+Verb+Pos+Imp+A2sg (belgeye el at-start working on a document)	0.04	0.56	0.69	0.76	0.73
R2	belge+Noun+A3sg+Pnon+Dat imza+Noun+A3sg+Pnon+Nom at+Verb+Pos+Imp+A2sg (belgeye imza at-sign a document)	0.04	0.61	0.78	0.76	1.0
R3	belge+Noun+A3sg+Pnon+Dat el+Noun+A3sg+Pnon+Nom koy+Verb+Pos+Imp+A2sg (belgeye el koy-seize a document)	0.31	0.56	0.66	0.76	0.39
R4	belge+Noun+A3sg+Pnon+Dat imza+Noun+A3sg+Pnon+Nom koy+Verb+Pos+Imp+A2sg (belgeye imza koy-invalid)	0.31	0.61	0.71	0.76	0.73

As we can see from the test results in Table 15 and Table 16, statistical language model is valuable and it can be used with an example based translation system. According to the results in Table 1, LM2 performs better for coefficients α=0.9$ and β=0.1$. Similar results can be observed in Table 16 for two coefficient pairs. This means that our statistical language model LM2 and our example-based translation part cooperate and they give the best result. In overall results, LM2 shows better performance than others. LM3 does not give too much information since there are much more equivalent tags of words in LM3. Also, LM1 is limited, since a word with its final tag can be encountered rarely in corpora. As we mentioned before LM1 is a SLM with words with their final tags, LM2 is a SLM with words with their root types, and LM3 is a SLM with final tags of words without their roots. Table 17 gives four possible Turkish

translations of a Kazakh sentence. For each translation result, its confidence factor, its language model probabilities for three language models (LM1, LM2 and LM3) and its BLEU score are given. Since the BLEU score of R2 is 1, it is the correct translation. According to their CFs, R3 and R4 have best scores, and one of them will be selected if the only EBMT part of our hybrid system is used. Since R2 which is the correct translation has highest score in LM2, LM2 can pick the correct answer.

Table 18. Enhanced results according to slms of Kazakh

#	**Source Sentence:** oğlan+Noun+A3pl+Pnon+Nom git+Verb+Neg+Prog1+A3pl (oğlanlar gitmiyorlar-boys aren't going) **Reference Sentence:** балалар бара жатқан жоқ	CF	LM1	LM2	LM3	BLEU
R 1	бала+Noun+A3pl+Pnon+Nom бар+Verb+Pos^DB+Adverb+ByDoningSo жат+Verb+Pos^DB+Noun+PastPart+A3sg+Pnon+Nom жоқ+Noun+A3sg+Pnon+Nom^DB+Verb+Zero+Pres+A 3sg (балалар бара жатқан жоқ -boys aren't going)	0.09	0.28	0.34	0.43	1.0
R 2	бала+Noun+A3pl+Pnon+Nom бар+Verb+Pos^DB+Adverb+AfterDoningSo жат+Verb+Pos^DB+Noun+PastPart+A3sg+Pnon+Nom жоқ+Noun+A3sg+Pnon+Nom^DB+Verb+Zero+Pres+A 3sg (балалар барып жатқан жоқ -boys aren't arriving)	0.33	0.18	0.23	0.43	0.86
R 3	бала+Noun+A3pl+Pnon+Nom кет+Verb+Pos^DB+Adverb+ByDoningSo жат+Verb+Pos^DB+Noun+PastPart+A3sg+Pnon+Nom жоқ+Noun+A3sg+Pnon+Nom^DB+Verb+Zero+Pres+A 3sg (балалар кете жатқан жоқ -invalid)	0.2	0.08	0.14	0.43	0.86
R 4	бала+Noun+A3pl+Pnon+Nom кет+Verb+Pos^DB+Adverb+AfterDoningSo жат+Verb+Pos^DB+Noun+PastPart+A3sg+Pnon+Nom жоқ+Noun+A3sg+Pnon+Nom^DB+Verb+Zero+Pres+A 3sg (балалар кетіп жатқан жоқ -boys aren't leaving) not common	0.27	0.08	0.14	0.43	0.76

R2 and R4 have highest score in LM1 and it cannot pinpoint the correct answer directly. Since LM3 produces same score for all translation results, it cannot make any judgment among results. We can say that LM2 performs better for

this example. Although Turkish and Kazakh are closely related languages, there are considerably many differences between them. For example, four different Kazakh sentences **“бара жатқан жоқ”**, **“бармай жатыр”**, **“кетпей жатыр”** and **“кетіп жатқан жоқ”** mean **“gitmiyorlar”** (they are not going) in Turkish with some semantic differences. They differ according to continuity and physical position of the doer while performing that process. In Kazakh language, there are 4 auxiliary verbs that are added to main verbs by indicating continuity and the physical position of the doer. These auxiliary verbs are **“тұр”** (standing by), **“жүр”** (walking), **“отыр”** (sitting) and **“жатыр”** (lying down). Thus, every verb in Kazakh language can be different in meaning depending on which auxiliary verb added to that verb. Therefore, for every Turkish verb in progressive tense, there are at least four kinds of its usage in Kazakh language. In order to find the valid one, confidence factors of learned translation templates may not be satisfactory, and we may also need information in a SLM. In our system, we have such kind of valuable ranking mechanism according to probabilities of translation results. In Table 18, Kazakh translation results of a Turkish sentence in progressive tense are given. The correct translation result is R1 and it can be captured by LM1 and LM2 since they have higher scores for R1. According to CFs, R1 has the lowest value and R2 has the highest value. LM3 again does not produce satisfactory results.

9. CONCLUSION AND FUTURE WORK

Nowadays there are significant research efforts on machine translation field with increased number of developing technologies. We aimed to make a contribution to this topic by trying to enhance translation results. For this reason, hybrid methodology is used and focused on combining the best properties of more than one MT approaches to provide better translation quality. So, we have worked out a hybrid system which is constituent of EBMT as core model and RBMT, SMT as information part of the system.

We examined all properties of MT approaches and their underlying problems, then decided that EBMT is best suited to language pair involved in the system. Furthermore, with these language pair we can get corpora which meet sublanguage criterion. Thus, in our hybrid system, translations are performed by the EBMT part of the system which is a core of the model. Since previously compiled and rule structured database are used and as far as it is linguistic part of the system, we can say that RBMT is used on information base to feed our system with structured and disambiguated data.

A valueable work which is studying Kazakh language entirely from computer linguistic view is performed for developing some stand-alone systems which are not worked out earlier such as a morphological processor and a morphological disambiguator of Kazakh language. As mentioned before, the languages involved in the system are agglutinative languages and morphology plays an important role in the structures of their words and their sentences structures. Thus, the morphological processors are used obtain the lexical forms of the surface level words and the surface level forms of translation results at lexical level. Also, translation templates are kept at lexical level and they translate a given source language sentence at lexical level to a target language sentence at lexical level.

Moreover, lessening ambiguity of a text before translation process is vital to increase the quality of translation results. Thus, to disambiguate words morphologically, a Turkish morphological disambiguator and a Kazakh morphological disambiguator are used.

However, as for future work, in order to get more accurate results, first of all disambiguation of Kazakh side should be worked out to better performance. In our system, it produces 89% accuracy and it should be enhanced up to 98% by adding some hand crafted rules. The improvement in disambiguation process will definitely have a positive effect in our translation system.

As the main part of our work, translation process had been performed bi-directionally and can translate new sentences in both directions. This means that a Kazakh sentence can be translated into Turkish and a Turkish sentence can be translated into Kazakh language using learned templates. Left sides of learned templates define a grammar of Kazakh language and right sides define a Turkish grammar. The translation process can be thought as a grammar correctness check of a given source language sentence by the defined source language grammar through learned templates and at the same time generating its translation in target language.

After results obtained from translation process, valid results are chosen according to confidence factors of translation results and SLMs of the target language. We have bilingual corpora of examples and confidence factor is calculated to choose valid results according to given database.

Although Turkish and Kazakh languages are closely related they have distinct features to be dealed in their domain. In other words, to get the valid result in target language, confidence factors of learned translation templates may not be satisfactory, and we may also need information in a statistical language model. Thus, combining example-based translation system with monolingual statistical language model produces some valuable results according to our experiments. In our system, we have such kind of valuable ranking mechanism according to probabilities of translation results. Our experiments show that we can rely on the combination of EBMT and SMT approaches, because it produces best results.

Although our language resources have reasonable sizes, they are not big as language resources used in other translation systems. We believe that the performance of our system will be enhanced if language resources are increased. Moreover, in order to catch common usage of results in target

language the corpora of SLM should be considerably large and should be increased up to millions level. These adjustments will make our system to produce more valuable and precise translation results.

REFERENCES

[1] W. N. Locke and A. D. Booth, “Machine translation of languages,” *Am. Doc.*, vol. 7, no. 2, pp. 135–136, **1956**.

[2] B. Vauquois, “A survey of formal grammars and algorithms for recognition and transformation in mechanical translation,” in *{IFIP} Congress {(2)}*, pp. 1114–1122, **1968.**

[3] W. J. Hutchins, “Chapter 6 Machine translation: History of Research and Applications” , **2004**.

[4] H. L. Somers, “Review Article: Example-based Machine Translation.,” *Machine Translation*, vol. 14, no. 2, pp. 113–157, **1999**.

[5] H. L. Somers, “Current research in Machine Translation,” *Machine Translation*, vol. 7, no. 4, pp. 231–246, **1992**.

[6] S. W. Chan, *Routledge Encyclopedia of Translation Technology*. Taylor & Francis, **2014**.

[7] A. Bharati, V. Chaitanya, A. P. Kulkarni, and R. Sangal, “Anusaaraka: Machine Translation in Stages,” *CoRR*, vol. cs.CL/0306, **2003**.

[8] D. J. Arnold, S. Krauwer, M. Rosner, L. des Tombe, and G. B. Varile, “The {\textless}C, A{\textgreater}, {T} Framework in Eurotra: {A} Theoretically Committed Notation for {MT},” in *Proceedings of the 11th International Conference on Computational Linguistics, {COLING} ’86, Bonn, Germany, August 25-29, 1986*, pp. 297–303, **1986**.

[9] B. J. Dorr, “A Parameterized Approach to Integrating Aspect with Lexical-Semanics for Machine Translation,” in *30th Annual Meeting of the Association for Computational Linguistics, 28 June - 2 July 1992, University of Delaware, Newark, Deleware, USA, Proceedings.*, pp. 257–264, **1992**.

[10] M. Nagao, “A framework of a mechanical translation between Japanese and English by analogy principle.,” *Elsevier Science*, **1984.** [Online]. Available: http://mt-archive.info/Nagao-1984.pdf. [Accessed: 10-Feb-2016].

[11] P. F. Brown, J. Cocke, S. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin, “A Statistical Approach to Machine Translation.,” *Comput. Linguist.*, vol. 16, no. 2, pp. 79–85, **1990**.

[12] “NTT Communication Science Laboratories Innovative Communication Laboratory.” [Online]. Available: http://www.kecl.ntt.co.jp/rps/english/Research_e/cn20/research_innovative03_e.html. [Accessed: 05-Apr-**2016**].

[13] P. Koehn, F. J. Och, and D. Marcu, “Statistical Phrase-Based Translation,” in *HLT-NAACL*, **2003**.

[14] O. Bender, K. Macherey, F. J. Och, and H. Ney, “Comparison of Alignment Templates and Maximum Entropy Models for {NLP},” in *{EACL} 2003, 10th Conference of the European Chapter of the*

Association for Computational Linguistics, April 12-17, 2003, Agro Hotel, Budapest, Hungary, pp. 11–18, **2003**.

[15] J. Riesa, A. Irvine, and D. Marcu, “Feature-Rich Language-Independent Syntax-Based Alignment for Statistical Machine Translation,” in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, {EMNLP} 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, {A} meeting of SIGDAT, a Special Interest Group of the {ACL}*, pp. 497–507, **2011**.

[16] D. Marcu, W. Wang, A. Echihabi, and K. Knight, “{SPMT:} Statistical Machine Translation with Syntactified Target Language Phrases,” in *{EMNLP} 2007, Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, 22-23 July 2006, Sydney, Australia*, pp. 44–52, **2006**.

[17] D. Liu and D. Gildea, “Can Semantic Roles Improve Syntax-Based Machine Translation?,” **2008**.

[18] P. Koehn and K. Knight, “Feature-Rich Statistical Translation of Noun Phrases,” in *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan.*, pp. 311–318, **2003**.

[19] D. S. Munteanu and D. Marcu, “Improving Machine Translation Performance by Exploiting Non-Parallel Corpora,” *Comput. Linguist.*, vol. 31, no. 4, pp. 477–504, **2005.**

[20] J. Hutchins, “Example-based machine translation: a review and commentary,” *Mach. Transl.*, vol. 19, no. 3–4, pp. 197–211, **2005.**

[21] H. L. Somers, *Recent Advances in Example-Based Machine Translation*, vol. 21. Dordrecht: Springer Netherlands, **2003.**

[22] E. Sumita and H. Iida, “Experiments and Prospects of Example-Based Machine Translation,” in *29th Annual Meeting of the Association for Computational Linguistics, 18-21 June 1991, University of California, Berkeley, California, USA, Proceedings.*, pp. 185–192, **1991**.

[23] H. Somers, “SOME THOUGHTS ON INTERFACE STRUCTURE(S).”

[24] I. Cicekli and H. A. Güvenir, “Learning Translation Templates from Bilingual Translation Examples,” *Appl. Intell.*, vol. 15, no. 1, pp. 57–76, Jun. **2001**.

[25] R. D. Brown, “Context-sensitive retrieval for example-based machine translation,” in *In Proceedings of Workshop: Example-Based Machine Translation, The Tenth Machine Translation Summit*, pp. 12–16, **2005**.

[26] H. Watanabe, S. Kurohashi, and E. Aramaki, “Finding Structural Correspondences from Bilingual Parsed Corpus for Corpus-based Translation,” in *{COLING} 2000, 18th International Conference on Computational Linguistics, Proceedings of the Conference, 2 Volumes, July 31 - August 4, 2000, Universit{ä}t des Saarlandes, Saarbr{ü}cken, Germany*, pp. 906–912, **2000**.

[27] C. Quirk and A. Menezes, “Dependency treelet translation: the convergence of statistical and example-based machine-translation?,” *Mach. Transl.*, vol. 20, no. 1, pp. 43–65, **2006**.

[28] S. Sato, “{CTM:} An Example-Based Translation Aid System,” in *14th International Conference on Computational Linguistics, {COLING} 1992, Nantes, France, August 23-28, 1992*, pp. 1259–1263, **1992**.

[29] L. Cranias, H. Papageorgiou, and S. Piperidis, “A Matching Technique In Example-Based Machine Translation,” in *15th International Conference on Computational Linguistics, {COLING} 1994, Kyoto, Japan, August 5-9, 1994*, pp. 100–104, **1994**.

[30] H. Maruyama and H. Watanabe, “Tree Cover Search Algorithm for Example-Based Translation Tree-cover search algorithm Problem definition,” *IBM Res. Tokyo Res. Lab.*, no. c, pp. 173–184, **1992**.

[31] S. Nirenburg, C. Domashnev, and D. J. Grannes, “Two Approaches to Matching in Example-Based Machine Translation,” in *In Proc. of the 5th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-93*, pp. 47–57, **1993**.

[32] B. Collins, “Example-Based Machine Translation: An Adaptation-Guided Retrieval Approach.” University of Dublin, Trinity College. Department of Computer Science, 07-Jun-**2006**.

[33] M. Nagao, J. Tsujii, K. Mitamura, H. Hirakawa, and M. Kume, “A Machine Translation System From Japanese Into English - Another Perspective Of MT Systems.,” in *COLING*, pp. 414–423, **1980**.

[34] S. Sato, “{MBT2:} {A} Method for Combining Fragments of Examples in Example-Based Translation,” *Artif. Intell.*, vol. 75, no. 1, pp. 31–50, **1995**.

[35] H. Watanabe, “A Similarity-Driven Transfer System,” in *14th International Conference on Computational Linguistics, {COLING} 1992, Nantes, France, August 23-28, 1992*, pp. 770–776, **1992**.

[36] H. Watanabe, “A model of a bi-directional transfer mechanism using rule combinations,” *Mach. Transl.*, vol. 10, no. 4, pp. 269–291, **1995**.

[37] Z. Öz and I. Cicekli, “Ordering Translation Templates by Assigning Confidence Factors,” pp. 51–61, Oct. **1998**.

[38] M. Carl, C. Pease, L. L. Iomdin, and O. Streiter, “Towards a Dynamic Linkage of Example-based and Rule-based Machine Translation.,” *Mach. Transl.*, vol. 15, no. 3, pp. 223–257, **2000**.

[39] M. R. Costa-Juss and Fonollosa, “Latest trends in hybrid machine translation and its applications,” *Comput. Speech Lang.*, vol. 32, no. 1, pp. 3–10, **2015**.

[40] K. R. Beesley and L. Karttunen, *Finite State Morphology.*, vol. 3. CSLI Publications, **2003**.

[41] O. Istek and I. Cicekli, “A link grammar for an agglutinative language.”

[42] K. Oflazer, “Two-level description of Turkish morphology,” in *Proceedings of the sixth conference on European chapter of the Association for Computational Linguistics*, p. 472, **1993**.

[43] K. Altintas and I. Cicekli, “A Morphological Analyser for Crimean Tatar,” Jan. **2001**.

[44] A. C. Tantuğ, E. Adalı, and K. Oflazer, *Advances in Natural Language Processing*, vol. 4139. Berlin, Heidelberg: Springer Berlin Heidelberg, **2006**.

[45] M. Orhun, A. Cuneyd Tantug, E. Adali, and A. C. Sonmez, “Computational comparison of the Uyghur and Turkish Grammar,” in *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pp. 343–347, **2009**.

[46] Ç. Çöltekin, “A Freely Available Morphological Analyzer for Turkish.,” in *LREC*, **2010**.

[47] G. Altenbek and R. Sun, “Kazakh Noun Phrase Extraction Based on N-gram and Rules,” in *2010 International Conference on Asian Language Processing*, pp. 305–308, **2010**.

[48] B. M. Kairakbay and D. L. Zaurbekov, “Finite State Approach to the Kazakh Nominal Paradigm.,” in *FSMNLP*, pp. 108–112, **2013**.

[49] J. Washington, I. Salimzyanov, and F. M. Tyers, “Finite-state morphological transducers for three Kypchak languages.,” in *LREC*, pp. 3378–3385, **2014**.

[50] O. Makhambetov, A. Makazhanov, I. Sabyrgaliyev, and Z. Yessenbayev, *Computational Linguistics and Intelligent Text Processing*, vol. 9041. Cham: Springer International Publishing, **2015**.

[51] D. R. Cutting, J. Kupiec, J. O. Pedersen, and P. Sibun, “A Practical Part-of-Speech Tagger.,” in *ANLP*, pp. 133–140, **1992**.

[52] H. Sak, T. Güngör, and M. Saraclar, *Computational Linguistics and Intelligent Text Processing*, vol. 4394. Berlin, Heidelberg: Springer Berlin Heidelberg, **2007.**

[53] D. Z. Hakkani-Tür, K. Oflazer, and G. Tür, “Statistical Morphological Disambiguation for Agglutinative Languages.,” *Comput. Hum.*, vol. 36, no. 4, pp. 381–410, **2002**.

[54] K. Oflazer and I. Kuruöz, “Tagging and Morphological Disambiguation of Turkish Text.,” in *ANLP*, pp. 144–149, **1994**.

[55] K. Oflazer and G. Tür, “Morphological Disambiguation by Voting Constraints.,” in *ACL*, pp. 222–229, **1997**.

[56] T. Daybelge and I Cicekli, “A Rule-Based Morphological Disambiguator for Turkish”, Recent Advances in Natural Language Processing - RANLP, pp. 145--149, **2007**

[57] E. Brill, “Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging.”

[58] P. Tapanainen and A. Voutilainen, “Tagging accurately - Don’t guess if you know.,” in *ANLP*, pp. 47–52, **1994**.

[59] M. Kutlu and I. Cicekli, “A Hybrid Morphological Disambiguation System for Turkish,” in *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pp. 1230–1236, **2013**.

[60] I. Cicekli, “Inducing translation templates with type constraints,” *Machine Translation*, vol. 19, no. 3–4, pp. 283–299, Jan. **2007**.

[61] A. C. Tantug, E. Adali, and K. Oflazer, *Advances in Natural Language Processing*, vol. 4139. Berlin, Heidelberg: Springer Berlin Heidelberg, **2006**.

[62] I. Washington, Jonathan North: Salymjanov, “A free / open-source Kazakh-Tatar machine translation system,” pp. 175–182, **2013**.

[63] I. Cicekli, “Inducing translation templates with type constraints,” *Mach. Transl.*, vol. 19, no. 3–4, pp. 283–299, **2005**.

[64] H. A. Güvenir and I. Cicekli, “Learning translation templates from examples,” *Inf. Syst.*, vol. 23, no. 6, pp. 353–363, Sep. **1998**.

[65] G. Kessikbayeva and I. Cicekli, “Rule Based Morphological Analyzer of Kazakh Language,” in *Proceedings of the 2014 Joint Meeting of SIGMORPHON and SIGFSM*, pp. 46–54, **2014**.

[66] M. Hulden, “Foma: a finite-state compiler and library,” pp. 29–32, Apr. **2009**.

[67] L. Anthony, “AntConc (Version 3.2. 2),” *Tokyo, Japan Waseda Univ.*, **2011**.

[68] T. Daybelge and I. Cicekli, “A ranking method for example based machine translation results by learning from user feedback,” *Appl. Intell.*, vol. 35, no. 2, pp. 296–321, Mar. **2010**.

[69] M. Silberztein, *Automata Implementation*, vol. 1436. Berlin, Heidelberg: Springer Berlin Heidelberg, **1998**.

[70] M. Mohri, F. Pereira, and M. Riley, “The design principles of a weighted finite-state transducer library,” *Theor. Comput. Sci.*, vol. 231, no. 1, pp. 17–32, Jan. **2000**.

[71] L. Karttunen, R. M. Kaplan, and A. Zaenen, “Two-Level Morphology with Composition.,” in *COLING*, pp. 141–148, **1992**.

[72] E. L. Antworth, “Glossing text with the PC-KIMMO morphological parser,” *Comput. Hum.*, vol. 26, no. 5–6, pp. 389–398, Dec. **1992**.

[73] G. D. Ritchie, “Languages Generated by Two-Level Morphological Rules.,” *Comput. Linguist.*, vol. 18, no. 1, pp. 41–59, **1992**.

[74] K. Demirci, *Kazakh verbal structures and descriptive verbs /*. Hyattsville: Dunwoody Press, **2006**.

[75] T. Valyaeva, “Казахский язык. Грамматика. Просто о сложном.” [Online]. Available: http://www.kaz-tili.kz/.

[76] E. Mamanov, I, “Қазақ тілі - Все для студента,” **1961**.

[77] K. Mussayev M, “Казахский язык - Все для студента,” **2008** .

[78] С. Исаев, *Сопоставительная типология казахского и русского языков*. Санат, **1996**.

[79] Ә Қараев, “Қазақ тілі,” **1993** .

[80] “KAZINFORM.” [Online]. Available: http://www.kazinform.kz/kaz.

[81] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, **2009**.

[82] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a Method for Automatic Evaluation of Machine Translation.,” in *ACL*, pp. 311–318, **2002**.

[83] E. M. Voorhees, “The TREC-8 Question Answering Track Report.,” in *TREC*, **1999**.

APPENDIX 1: ORTHOGRAPHIC RULES OF KAZAKH LANGUAGE

```
define FrontVowel [ і | ұ | ө | ә | е | и | Е] ;
define BackVowel [ ы | ұ | о | а | я | О] ;
define Vowel [ а | ә | о | ө | ұ | ұ | ы | і | е | и | ю | я ] ;
define Cons [ б | в | г | ғ | д | ж | з | й | к | қ | л | м | н | ң | п | р | с | т | ф | х | һ | ц | ч | ш | щ ] ;
define ConsWOPL [ б | в | г | ғ | д | ж | з | к | қ | м | ң | п | с | т | ф | х | һ | ц | ч | ш | щ ] ;
define VCons1 [ ж | з ] ;
define VCons2 [ б |в | г | ғ | д ] ;
define VCons [ ж | з | б | в | г | ғ | д ] ;
define SonCons1 [ л | р | у | й ] ;
define SonCons2 [ м | н | ң] ;
define SonConsw [ л | р | й |м | н | ң ] ;
define SonCons [ м | н | ң | л | р | й | у ] ;
define ConsWI [ к | с | т | б | ш ] ;
define ConsWOI [ ж | қ ] ;
define UnVCons [ к | қ | п | с | т | ш | щ | ф | х | һ | ц | ч ] ;
define Part1 [Vowel|SonCons1|VCons1] ;
define Part2 [Vowel|SonCons|VCons] ;
define Part3 [SonCons1|VCons1] ;
define Part4 [SonCons|VCons1] ;
read lexc all.lexc
define Lexicon
define OERule01     и -> Е || ?*_(з)[м]"$";
define OERule02     и -> Е || ?*_[в|ф]"$";
define OERule0      и -> Е || ?*_[ConsWI]"$";
define OERule1      и -> О || [BackVowel]?*_?*"$";
define OERule2      и -> Е || [FrontVowel]?*_?*"$";
define OERule3      и -> {Ей} || .#.[ConsWI]_ ,  .#._ ;
define OERule4      и -> {Ой} || .#.[ConsWOI]_ ,  .#._ , ?*_[я]"$";
define AYURule1     ю -> {йу} || .#.[Vowel]_ ;
define URule        у -> {у#} || .#.(Cons)*_(Cons)*"$",.#.?*[BackVowel](Cons)*_"$";
define HRule1       Н -> 0 || [Vowel|J|A]"$"[1|2|0]_Cons ;
define HRule2       Н -> J ;
define BRule1       В -> 0 || [Vowel|J|A]"$"[1|2|0]_[Cons] ;
define BRule2       В -> А ;
define WRule        Vowel -> 0 || _"$"[W];    #XAw ekew
define ARule        А -> е , J -> і || ["&"]"$" ?*_ ,,  "&" -> 0 ;
define VHRule       А -> й || [Vowel|A|J]"$"_ ;
define VARule       А -> а , J -> ы ||[BackVowel|"#"](у)(Cons)*"$"?*_ ,[Cons]у[Cons]*"$"?*_;
define Cleanup1      "#" -> 0 ;
define VERule       А -> е , J -> і |||[FrontVowel](у)(Cons)*"$"?*_  ;
define PossRule1     с -> 0 || [К|Cons|у]"$"[3]_ ;
define BtoWRule      б -> у || [Vowel]_"$"[1|2|3|"@"|0][ы|і][п] ;
define LocRule      С -> {нд} ||"$"[3]?*[ ы | і ]"$"_[5] ;
define AblRule1     С -> н || [SonCons2]"$"_[4],"$"[3]?*[ы|і]"$"_[4] ;
define AblRule2     С -> д || [Part2]"$"_[5],[Part1]"$"_[4] ;
define DatRule1     С -> н || "$"[3]?*[ ы | і ]"$"_[6] ;
define DatRule2     С -> 0 || "$"[1|2]?*_[6] ;
define DatRule3     С -> Т || [Part2] "$"_[6],[UnVCons|К]"$"_[6];
define AccRule1     С -> н || [Vowel]"$"_[8] ;
define AccRule2     ы -> 0 , і ->0 || "$"[3]?*[8]_ ;
define GenRule1      С -> н || [Vowel|SonCons2]"$"_[7] ;
```

```
define PossAffRule1   C -> н || [Vowel] "$"_{88} ;
define PossAffRule2   C -> д || [Part4] "$"_{88},[Part4] "$"_[8],[Part3] "$"_[7] ;
define RelRule1       R -> ᵻ || [BackVowel]"$"_ ;
define RelRule2       R -> г || [FrontVowel]"$"_ ;
define InsRule1       C -> б || [VCons1] "$"[1|0]_[9] ;
define InsRule2       C -> п || [К|UnVCons]"$"[1|0]_[9] ;
define InsRule3       C -> м || [Vowel|SonCons]"$"[1|0]_[9],"$"[3][I | i]_[9] ;
define Cleanup3        4 -> 0, 5 -> 0, 6 -> 0, 7 -> 0, 8 -> 0, 9 -> 0 ;
define PluRule0       P -> 0 || [з][д]"$"_ ;
define PluRule1       P -> д || [SonCons2| л |VCons1] "$"_ ;
define PluRule2       P -> л || [Vowel|й|р|у]"$"_ ;
define PluRule3       P -> т, С -> т || [К|VCons2|UnVCons]"$"_ ;
define YARule1        а -> я || [О|Е|й]"$"_?* ;
define YARule2        й -> 0 || _"$"[я]?* ;
define YIRule1        ы->U, i->U || [Cons]_"$"[у]?* ;
define YURule1        у -> ю || [О|Е|i|ы|й]["$"|0]_(?)* ;
define YURule2        й -> 0, i -> 0, ы -> 0 || _[i|ы|0]["$"|0][ю](?)* ;
define JRule1        ы -> и , i -> и || _"$"[1|0][й]?* ;
define JRule2        й -> 0 || [и]"$"[1|0]_?* ;
define QRule1         Q -> г || [Vowel|SonCons]"$"_ ;
define QRule2         Q -> к || (UnVCons)"$"_ ;
define KRule1         К -> ᵸ || [BackVowel]?*"$"(?*)[BackVowel|1]_ ;
define KRule2         К -> к || FrontVowel]?*"$"(?*)[FrontVowel|1]_ ;
define SoftRule       к -> г , ᵸ -> ᵻ , п -> б || [Vowel]_"$"[1|2|3|"@"|0][Vowel|у] ;
define Cleanup2        "$" -> 0 || _ [у]"$";
define TRule1         Т -> ᵸ || [BackVowel|у](?*)[UnVCons]"$"_[BackVowel] ;
define TRule2         Т -> к || [FrontVowel](?*)[UnVCons]"$"_[FrontVowel] ;
define TRule3         Т -> ᵻ || [BackVowel](?*)(Cons)[0|SonCons|VCons1|ю]"$"_ ,
                     .#.[Cons](у)*"$"_ ;
define TRule3         Т -> ᵻ || [BackVowel](?*)(Cons)[0|SonConsw|VCons1|ю]"$"_ ,
                     .#.[Cons](у)*"$"_ ;
define TRule4         Т -> г || [FrontVowel](?)(?)(?)(Cons)[0|SonConsw|VCons1]"$"_ , "$"_[FrontVowel];

define DRule1         D -> д || [SonCons|VCons|Vowel]("$")"$"[3|0]_ ;
define DRule2         D -> т || [UnVCons] "$"[3|0]_ ;
define NegRule1       М -> б || [VCons1|SonCons2]"$"_ ;
define NegRule2       М -> п || [UnVCons]"$"_ ;
define NegRule3       М -> м || [Vowel|SonCons1]"$"_ ;
define CleanOERule    О -> и, {Ей}-> и, {Ой}-> и, Е -> и ;
define OneSyll        р -> {рZ} ||.#.(Cons)[Vowel]_"$";
define Caus           т -> 0 || [ConsWOPL | Z]"$"[V]_ ;
```

APPENDIX 2: PART OF LEXICON FOR KAZAKH LANGUAGE

LEXICON Root	
	PRONOUN;
	NOUN;
	POSTPOSITION;
	CONJUNCTION;
	NUMERAL;
	ADJECTIVE;
	ADVERB;
	VERB;
	PUNCTUATION;
LEXICON PRONOUN	
	PERSON-PRON;
	INDICATIVE-PRON;
	INTERROGATIVE-PRON;
	DEFINITE-PRON;
	INDEFINITE-PRON;
	NEGATIVE-PRON;
	REFLEXIVE-PRON;
LEXICON PERSON-PRON	
мен+Pron+PersP+A1sg:мен	PERS-POSS1;
сен+Pron+PersP+A2sg:сен	PERS-POSS1;
сіз+Pron+PersP+A2psg:сіз	PERS-POSS2;
ол+Pron+PersP+A3sg+Pnon:он	PERS-STATE2;
біз+Pron+PersP+A1pl:біз	PERS-POSS2;
сендер+Pron+PersP+A2pl:сендер	PERS-POSS2;
сіздер+Pron+PersP+A2ppl:сіздер	PERS-POSS2;
мен+Pron+PersP+A1sg+Pnon+Nom:мен	DERIV;
сен+Pron+PersP+A2sg+Pnon+Nom:сен	DERIV;
сіз+Pron+PersP+A2psg+Pnon+Nom:сіз	DERIV;
ол+Pron+PersP+A3sg+Pnon:он	DERIV;
біз+Pron+PersP+A1pl+Pnon+Nom:біз	DERIV;
сендер+Pron+PersP+A2pl+Pnon+Nom:сендер	DERIV;
сіздер+Pron+PersP+A2ppl+Pnon+Nom:сіздер	DERIV;
олар+Pron+PersP+A3pl:олар	PERS-POSS2;
мен+Pron+PersP+A1sg:маған	PERS-POSS3;
сен+Pron+PersP+A2sg:саған	PERS-POSS3;
ол+Pron+PersP+A3sg:оған	PERS-POSS3;
LEXICON PERS-POSS1	
+Pnon:0	PERS-CASE1;
LEXICON PERS-POSS2	
+Pnon:0	CASE-ALL;
LEXICON PERS-POSS3	
+Pnon:0	PERS-CASE2;

LEXICON PERS-CASE1	
+Nom:0	FINAL;
+Nom:0	DERIV0;
+Loc:$C5A	STATE1;
+Gen:$Jң	STATEGEN;
+Ins:$Jмен	FINAL;
	PERS-STATE2;
LEXICON PERS-CASE2	
+Dat:0	FINAL;
LEXICON PERS-STATE2	
+Acc:$J	FINAL;
^DB+Pron+Poss+Rel:$ікі&	FINAL;
^DB+Pron+Poss+Rel^DB+Noun+Zero:$ікі&	NOUN-SUFF;
LEXICON REFLEXIVE-PRON	
өз+Pron+RefP:өз	REF-POSS;
LEXICON REF-POSS	
+A1sg+Pnon+Nom:0	FINAL;
+A1sg+P1sg:$1Нм	CASE-ALL;
+A2sg+P2sg:$2Нң	CASE-ALL;
+A2psg+P2psg:$НңJз	CASE-ALL;
+A3sg+P3sg:$3Н	CASE-ALL;
+A1pl+P1pl:$НмJз	CASE-ALL;
+A2pl+P2pl:PAp2Jң	CASE-ALL;
+A2ppl+P2ppl:PApJңJз	CASE-ALL;
+A3pl+P3pl:PAp3J	CASE-ALL;
LEXICON INDICATIVE-PRON	
бұл+Pron+IndcP+A3sg+Pnon+Nom:бұл	FINAL;
осы+Pron+IndcP+A3sg+Pnon+Nom:осы	FINAL;
сол+Pron+IndcP+A3sg+Pnon+Nom:сол	FINAL;
ол+Pron+PersP+A3sg+Pnon+Nom:ол	FINAL;
мынау+Pron+IndcP+A3sg+Pnon+Nom:мынау	FINAL;
осы+Pron+IndcP+A3sg+Pnon+Nom:осынау	FINAL;
мынау+Pron+IndcP+A3sg+Pnon+Nom:мына	FINAL;
анау+Pron+IndcP+A3sg+Pnon+Nom:анау	FINAL;
анау+Pron+IndcP+A3sg+Pnon+Nom:ана	FINAL;
мынау+Pron+IndcP+A3sg+Pnon+Nom:мұ	DERIVPR1;
бұл+Pron+IndcP+A3sg+Pnon+Nom:бұ	DERIVPR1;
осы+Pron+IndcP+A3sg+Pnon+Nom:осы	DERIVPR1;
осы+Pron+IndcP+A3sg+Pnon+Nom:о	DERIVPR1;
сол+Pron+IndcP+A3sg+Pnon+Nom:со	DERIVPR1;
мынау+Pron+IndcP+A3sg+Pnon+Nom:мына	DERIVPR2;
анау+Pron+IndcP+A3sg+Pnon+Nom:ана	DERIVPR2;
бұл+Pron+IndcP+A3sg+Pnon+Nom:бұ	FINAL;
осы+Pron+IndcP+A3sg+Pnon+Nom:о	FINAL;
бұл+Pron+IndcP:бұ	NOUN-AGR-SUFF-PRN1;
мынау+Pron+IndcP:мұ	NOUN-AGR-SUFF-PRN1;
осы+Pron+IndcP:осы	NOUN-AGR-SUFF-PRN2;
осы+Pron+IndcP:о	NOUN-AGR-SUFF-PRN1;
сол+Pron+IndcP:со	NOUN-AGR-SUFF-PRN1;
ол+Pron+PersP:о	NOUN-AGR-SUFF-PRN1;
мынау+Pron+IndcP:мына	NOUN-AGR-SUFF-PRN3;

анау+Pron+IndcP:ана	NOUN-AGR-SUFF-PRN3;
LEXICON NOUN-AGR-SUFF-PRN1	
+A3sg:0	POSS-AFFX3-PRN1;
+A3pl:$PAp	POSS-AFFX1;
LEXICON NOUN-AGR-SUFF-PRN2	
+A3sg:0	POSS-AFFX3-PRN2;
+A3pl:$PAp	POSS-AFFX1;
LEXICON NOUN-AGR-SUFF-PRN3	
+A3sg:0	POSS-AFFX3-PRN3;
+A3pl:$PAp	POSS-AFFX1;
LEXICON POSS-AFFX3-PRN1	
+Pnon:0	INDC-PRN-CASE1;
+P1sg:$1нJм	CASE-ALL;
+P2sg:$2нJң	CASE-ALL;
+P2psg:$@нJңJз	CASE-ALL;
+P3sg:$3cJ	CASE-ALL;
+P1pl:$@нJмJз	CASE-ALL;
+P2pl:PAp2Jң	CASE-ALL;
+P2ppl:PAp@JңJз	CASE-ALL;
+P3pl:PAp3J	CASE-ALL;
LEXICON POSS-AFFX3-PRN2	
+Pnon:0	INDC-PRN-CASE2;
+P1sg:$1Нм	CASE-ALL;
+P2sg:$2Нң	CASE-ALL;
+P2psg:$@НңJз	CASE-ALL;
+P3sg:$3cJ	CASE-ALL;
+P1pl:$@НмJз	CASE-ALL;
+P2pl:PAp2Jң	CASE-ALL;
+P2ppl:PAp@JңJз	CASE-ALL;
+P3pl:PAp3J	CASE-ALL;
LEXICON POSS-AFFX3-PRN3	
+Pnon:0	INDC-PRN-CASE3;
+P1sg:$1уJм	CASE-ALL;
+P2sg:$2уJң	CASE-ALL;
+P2psg:$@уJңJз	CASE-ALL;
+P3sg:$3уJ	CASE-ALL;
+P1pl:$@уJмJз	CASE-ALL;
+P2pl:PAp2Jң	CASE-ALL;
+P2ppl:PAp@JңJз	CASE-ALL;
+P3pl:PAp3J	CASE-ALL;
LEXICON INDC-PRN-CASE1	
+Dat:ған	FINAL;
+Abl:$дАн	FINAL;
+Gen:$нJң	FINAL;
+Acc:$нJ	FINAL;
+Loc:$нда	FINAL;
+Ins:$нJмен	FINAL;

LEXICON DERIVPR1	
^DB+Adj+INDPRN:$ндАй	ADJ-SUFF;
^DB+Adv+INDPRN:$лАй	ADV-SUFF;
^DB+Noun+INDPRN:$ндАғJ	NOUN-SUFF;
LEXICON DERIVPR2	
^DB+Adj+INDPRN:$ндАй	ADJ-SUFF;
^DB+Adj+INDPRN:$дАй	ADJ-SUFF;
^DB+Noun+INDPRN:$дАғJ	NOUN-SUFF;
^DB+Adv+INDPRN:$лАй	ADV-SUFF;
LEXICON INDC-PRN-CASE2	
+Dat:ған	FINAL;
+Abl:$дАн	FINAL;
+Gen:$нJң	FINAL;
+Acc:$нJ	FINAL;
+Loc:$нда	FINAL;
+Ins:$мен	FINAL;
LEXICON INDC-PRN-CASE3	
+Dat:ған	FINAL;
+Abl:$дАн	FINAL;
+Gen:$нJң	FINAL;
+Acc:$нJ	FINAL;
+Loc:$да	FINAL;
+Ins:$мен	FINAL;
LEXICON INTERROGATIVE-PRON	
кім+Pron+Intrg:кім	NOUN-AGR-SUFF-ALL;
не+Pron+Intrg:не	NOUN-AGR-SUFF-ALL;
қандай+Pron+Intrg:қандай	NOUN-AGR-SUFF-ALL;
қанша+Pron+Intrg:қанша	NOUN-AGR-SUFF-ALL;
қайсы+Pron+Intrg:қайсы	NOUN-AGR-SUFF-ALL;
неше+Pron+Intrg:неше	NOUN-AGR-SUFF-ALL;
қай+Pron+Intrg:қай	FINAL;
қайда+Pron+Intrg:қайда	NOUN-AGR-SUFF-ALL;
қашан+Pron+Intrg:қашан	FINAL;
қайдан+Pron+Intrg:қайдан	FINAL;
қалай+Pron+Intrg:қалай	FINAL;
LEXICON INDEFINITE-PRON	
біреу+Pron+Indef:біреу	NOUN-AGR-SUFF-ALL;
бірдеме+Pron+Indef:бірдеме	NOUN-AGR-SUFF-ALL;
әлдеқандай+Pron+Indef:әлдеқандай	NOUN-AGR-SUFF-ALL;
біраз+Pron+Indef:біраз	NOUN-AGR-SUFF-ALL;
кейбір+Pron+Indef:кейбір	NOUN-AGR-SUFF-ALL;
кейбір+Pron+Indef:кейбіреу	NOUN-AGR-SUFF-ALL;
бірнеше+Pron+Indef:бірнеше	NOUN-AGR-SUFF-ALL;
бірқатар+Pron+Indef:бірқатар	NOUN-AGR-SUFF-ALL;
әлдеқайда+Pron+Indef:әлдеқайда	NOUN-AGR-SUFF-ALL;
әлдеқашан+Pron+Indef:әлдеқашан	NOUN-AGR-SUFF-ALL;
әлдеқайдан+Pron+Indef:әлдеқайдан	FINAL;
кей+Pron+Indef:кей	FINAL;
LEXICON NEGATIVE-PRON	

ештеме+Pron+Negat:ештеме	NOUN-AGR-SUFF-ALL;
ештеңе+Pron+Negat:ештеңе	NOUN-AGR-SUFF-ALL;
ешнәрсе+Pron+Negat:ешнәрсе	NOUN-AGR-SUFF-ALL;
ешкім+Pron+Negat:ешкім	NOUN-AGR-SUFF-ALL;
ешқандай+Pron+Negat:ешқандай	NOUN-AGR-SUFF-ALL;
ешқайда+Pron+Negat:ешқайда	FINAL;
ешқашан+Pron+Negat:ешқашан	FINAL;
ешқалай+Pron+Negat:ешқалай	FINAL;
ешқайсы+Pron+Negat:ешқайсы	DEF-AGRMENT-POSS3;
ешбір+Pron+Negat:ешбір	DEF-AGRMENT-POSS1;
LEXICON DEFINITE-PRON	
бәрі+Pron+Def:бәр	DEF-AGRMENT-POSS1;
барлық+Pron+Def:барлық	DEF-AGRMENT-POSS2;
әрқайсы+Pron+Def:әрқайсы	DEF-AGRMENT-POSS3;
әрбір+Pron+Def:әрбір	DEF-AGRMENT-POSS2;
әркім+Pron+Def:әркім	DEF-AGRMENT-POSS2;
бүкіл+Pron+Def:бүкіл	DEF-AGRMENT-POSS2;
барша+Pron+Def:барша	DEF-AGRMENT-POSS2;
әрқашан+Pron+Def:әрқашан	FINAL;
әрқалай+Pron+Def:әрқалай	FINAL;
әрдайым+Pron+Def:әрдайым	FINAL;
әрқайда+Pron+Def:әрқайда	FINAL;
әр+Pron+Def:әр	FINAL;
LEXICON DEF-AGRMENT-POSS1	
+A3sg+P3sg:$3H	CASE-ALL;
+A1pl+P1pl:$HмJз	CASE-ALL;
+A2pl+P2pl:$HPAp$2Jң	CASE-ALL;
+A2ppl+P2ppl:$HPAp$JңJз	CASE-ALL;
+A3pl+P3pl:$HPAp$3J	CASE-ALL;
LEXICON DEF-AGRMENT-POSS2	
+A3sg+Pnon:0	CASE-ALL;
+A3sg+P3sg:$3H	CASE-ALL;
+A1pl+P1pl:$HмJз	CASE-ALL;
+A2pl+P2pl:PAp2Jң	CASE-ALL;
+A2ppl+P2ppl:PApJңJз	CASE-ALL;
+A3pl+P3pl:PAp3J	CASE-ALL;
LEXICON DEF-AGRMENT-POSS3	
+A3sg+Pnon:0	CASE-ALL;
+A3sg+P3sg:$3cJ	CASE-ALL;
+A1pl+P1pl:$HмJз	CASE-ALL;
+A2pl+P2pl:PAp2Jң	CASE-ALL;
+A2ppl+P2ppl:PApJңJз	CASE-ALL;
+A3pl+P3pl:PAp3J	CASE-ALL;
LEXICON NOUN	
абақты	N;
аберрация	N;
абсолют	N;
абырой	N;
LEXICON N	

+Noun:0	NOUN-SUFF;
LEXICON NOUN-SUFF	
	NOUN-AGR-SUFF-ALL;
LEXICON NOUN-AGR-SUFF-ALL	
+A3sg:0	POSS-AFFX2;
+A3pl:$PAp	POSS-AFFX1;
LEXICON POSS-AFFX1	
+Pnon:0	CASE-ALL;
+P1sg:$1Нм	CASE-ALL;
+P2sg:$2Нң	CASE-ALL;
+P2psg:$НңJз	CASE-ALL;
+P3sg:$3cJ	CASE-ALL;
+P1pl:$НмJз	CASE-ALL;
+P2pl:$2Нң	CASE-ALL;
+P2ppl:$НңJз	CASE-ALL;
+P3pl:$3cJ	CASE-ALL;
LEXICON POSS-AFFX2	
+Pnon:0	CASE-ALL;
+P1sg:$1Нм	CASE-ALL;
+P2sg:$2Нң	CASE-ALL;
+P2psg:$НңJз	CASE-ALL;
+P3sg:$3cJ	CASE-ALL;
+P1pl:$НмJз	CASE-ALL;
+P2pl:PAp2Нң	CASE-ALL;
+P2ppl:PApНңJз	CASE-ALL;
+P3pl:PAp3J	CASE-ALL;
LEXICON CASE-ALL	
+Nom:0	STATENOM;
+Ins:$C9ен&	STATEINS;
+Gen:$C7Jң	STATEGEN;
+Dat:$C6A	STATE0;
+Abl:$C4Aн	STATE0;
+Loc:$C5A	STATE1;
	STATE2;
LEXICON STATENOM	
	FINAL;
	DERIV;
	DERIV0;
	DERIV1;
LEXICON STATEINS	
	FINAL;
	DERIV;
LEXICON STATEGEN	
	FINAL;
	DERIV;

LEXICON STATE0	
	FINAL;
^DB+Adverb+Zero:0	FINAL;
LEXICON STATE1	
	DERIV0;
	FINAL;
^DB+Adj+Rel:$RJ	FINAL;
^DB+Noun+Rel:$RJ	NOUN-SUFF;
LEXICON STATE2	
+Acc:$C8J	FINAL;
+Acc:$C8J	DERIV;
^DB+Pron+Poss+Rel:$C88ікі&	FINAL;
^DB+Pron+Poss+Rel^DB+Noun+Zero:$C88ікі&	NOUN-SUFF;
LEXICON DERIV0	
^DB+Verb+Zero+Pres:0	AGREEMENT0;
^DB+Verb+Zero+Pres:$PAp	AGREEMENT0PL;
LEXICON DERIV1	
^DB+Adj+Zero:0	FINAL;
LEXICON DERIV	
^DB+Noun+Agt1:$шJ	NOUN-SUFF;
^DB+Noun+Small1:$шA	NOUN-SUFF;
^DB+Noun+Small2:$шJK	NOUN-SUFF;
^DB+Noun+Agt2:$Qep	NOUN-SUFF;
^DB+Noun+Theme:$PJK	NOUN-SUFF;
^DB+Noun+Group:$PAc	NOUN-SUFF;
^DB+Noun+Rel1:$тай	NOUN-SUFF;
^DB+Noun+Rel2:$й	NOUN-SUFF;
^DB+Adj+FitFor:$PJK	FINAL;
^DB+Noun+Place:$хана	NOUN-SUFF;
^DB+Adj+Group:$PAc	ADJ-SUFF;
^DB+Adj+Between:$аралық	ADJ-SUFF;
^DB+Adj+Agt2:$Qep	ADJ-SUFF;
^DB+Adj+Agt1:$шJ	ADJ-SUFF;
^DB+Adj+With:$PJ	ADJ-SUFF;
^DB+Adj+Without:$cJз	ADJ-SUFF;
^DB+Adj+qor:$қор	FINAL;
^DB+Adj+qoy:$қой	FINAL;
^DB+Adj+Time:$TJ	ADJ-SUFF;
^DB+Adj+Issue:$и	FINAL;
^DB+Adj+SeemAS:$шJл	FINAL;
^DB+Adj+As:$DАй	ADJ-SUFF;
^DB+Verb+Do:$PA	VERB-SUFF;
^DB+Verb+Perform:$PAc	VERB-SUFF;
^DB+Verb+SeemAs:$cJ	VERB-SUFF;
^DB+Adverb+AccordingTo1:$ншA	ADV-SUFF;
^DB+Adverb+AccordingTo2:$шA	ADV-SUFF;
^DB+Adverb+As1:$PAй	ADV-SUFF;
^DB+Adverb+As2:$DАйJн	ADV-SUFF;

APPENDIX 3: RULES LEARNED FOR DISAMBIGUATION

2 : Adverb+AdjMdfy : fors : Noun+A3sg+Pnon+Loc : correct = Adj
1 : Verb+Pos^DB+Adj+FutPart : fors : Punc : correct = Verb+Pos+Fut+A3sg
1 : Noun+A3pl+P3sg+Nom : pre : Verb+Pos^DB+Noun+Inf3+A3sg+Pnon+Nom Punc : correct = Noun+A3pl+P3pl+Nom
2 : Verb+Pos+Fut+A3sg : fors : Noun+A3sg+Pnon+Nom : correct = Adj
1 : Verb+Pos+Fut+A3sg : fors : Verb+Pos^DB+Adj+PresPart : correct = Verb+Pos^DB+Adj+FutPart
2 : Noun+A3sg+Pnon+Nom : fors : Pron+A1pl+Pnon+Dat Punc : correct = Verb+Pos+Aor+A3sg
1 : Verb^DB+Verb+Pass+Pos^DB+Adj+FutPart : fors : Punc : correct = Verb^DB+Verb+Pass+Pos+Fut+A3sg
2 : Adverb+Ques : fors : Noun+A3sg+Pnon+Nom : correct = Adj+Ques
2 : Adverb+AdjMdfy : fors : Noun+A3sg+Pnon+Nom Noun+A3sg+Pnon+Nom : correct = Adj
2 : Noun+A3sg+Pnon+Acc : fors : Postp : correct = Noun+A3sg+P3sg+Nom
2 : Adverb : fors : Noun+A3pl+P3sg+Loc : correct = Noun+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom^DB+Adj+FitFor : fors : Noun+A3sg+Pnon+Nom Noun+A3sg+P3sg+Dat : correct = Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom
2 : Interj : fors : Noun+Prop+A3sg+P3sg+Loc Num+Card : correct = Noun+Prop+A3sg+Pnon+Nom
2 : Verb+Pos+Fut+A3sg : fors : Noun+A3pl+Pnon+Loc Punc : correct = Adj
1 : Verb+Pos^DB+Adj+PresPart : pre : Noun+Prop+A3sg+Pnon+Loc : correct = Verb^DB+Verb+Pass+Pos^DB+Adj+PresPart
2 : Noun+A3sg+Pnon+Dat : pre : Noun+A3sg+Pnon+Nom Noun+A3pl+P3sg+Gen : correct = Noun+A3sg+P3sg+Dat
1 : Noun+A3sg+Pnon+Nom : pre : Noun+Prop+A3sg+Pnon+Nom Noun+A3sg+Pnon+Nom^DB+Adj+Agt : correct = Noun+Prop+A3sg+Pnon+Nom
2 : Adj : pre : Punc Num+Percent : correct = Noun+A3sg+Pnon+Nom
2 : Adj : fors : Punc Noun+A3sg+Pnon+Abl : correct = Pron+A3sg+Pnon+Nom
2 : Adj : fors : Postp Noun+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Dat
1 : Adj : fors : Verb^DB+Verb+Pass+Pos+Past+A3sg Punc : correct = Adverb
1 : Adj : fors : Verb^DB+Verb+Pass^DB+Verb+Able+Pos+Aor+A3sg Punc : correct = Pron+A3sg+Pnon+Nom
2 : Adverb+AdjMdfy : fors : Noun+A3sg+Pnon+Nom Verb+Pos+Aor+A3sg : correct = Adj
2 : Adverb+AdjMdfy : fors : Noun+A3sg+Pnon+Abl : correct = Adj
2 : Adverb+AdjMdfy : fors : Noun+A3sg+Pnon+Acc : correct = Adj
2 : Noun+A3sg+Pnon+Acc : pre : Noun+A3sg+Pnon+Nom Conj : correct = Noun+A3sg+P3sg+Nom
2 : Noun+A3sg+Pnon+Acc : fors : Noun+A3sg+P3sg+Loc : correct = Noun+A3sg+P3sg+Nom
1 : Adverb : fors : Noun+A3sg+P3sg+Nom : correct = Noun+A3sg+Pnon+Nom
2 : Postp : pre : Adverb+AdjMdfy Noun+Prop+A3sg+Pnon+Nom : correct = Adverb
2 : Postp : fors : Noun+A3sg+Pnon+Acc Noun+A3sg+P3sg+Nom : correct = Conj
2 : Postp : fors : Adj Noun+Prop+A3sg+Pnon+Gen : correct = Conj
2 : Postp : fors : Pron+Reflex+A3sg+P3sg+Nom Noun+A3sg+P3sg+Loc : correct = Conj
1 : Postp : pre : Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3pl+P3sg+Nom : correct = Conj
1 : Verb+Neg^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom : fors : Noun+A3sg+Pnon+Gen Adj : correct = Verb+Neg^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc
1 : Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Noun+Ness+A3pl+P3sg+Gen : fors : Noun+Prop+A3sg+Pnon+Gen Num+Ord : correct = Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Noun+Ness+A3pl+P3pl+Gen
1 : Verb+Pos^DB+Noun+Inf2+A3pl+P3sg+Nom : fors : Punc Adverb : correct = Verb+Pos^DB+Noun+Inf2+A3pl+P3pl+Nom
1 : Verb+Pos^DB+Noun+Inf2+A3sg+P3pl+Nom : fors : Adj^DB+Noun+Zero+A3sg+P3sg+Loc Punc : correct = Verb+Pos^DB+Noun+Inf2+A3pl+P3sg+Nom
1 : Verb+Neg^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc : pre : Noun+A3sg+P3sg+Acc Verb+Pos^DB+Adj+PresPart^DB+Noun+Zero+A3pl+Pnon+Gen : correct = Verb+Neg^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom

1 : Verb+Pos^DB+Noun+Inf2+A3sg+Pnon+Nom : fors : Noun+A3sg+Pnon+Nom Punc : correct = Verb+Neg+Imp+A2sg
2 : Verb+Pres+A1sg : pre : Adj Adj : correct = Noun+A3sg+P1sg+Nom
2 : Verb+Pos+Narr+A3sg : pre : Noun+A3sg+P3sg+Dat Noun+Prop+A3sg+Pnon+Nom : correct = Adj
2 : Noun+Ness+A3sg+P2sg+Gen : pre : Noun+A3sg+Pnon+Nom Adj : correct = Noun+Ness+A3sg+P3sg+Gen
1 : Noun+A3pl+P3pl+Loc : fors : Adj^DB+Noun+Zero+A3sg+Pnon+Dat Adj : correct = Noun+A3sg+P3pl+Loc
1 : Noun+A3sg+P3pl+Gen : fors : Noun+A3sg+P3sg+Gen Num+Card : correct = Noun+A3pl+P3sg+Gen
1 : Verb+Pos^DB+Adj+FutPart^DB+Noun+Zero+A3pl+P3pl+Acc : fors : Verb^DB+Verb+Caus+Pos^DB+Adverb+ByDoingSo Punc : correct = Verb+Pos^DB+Adj+FutPart^DB+Noun+Zero+A3pl+P3sg+Acc
1 : Verb+Pos^DB+Adj+Agt^DB+Noun+Zero+A3pl+P3pl+Nom : fors : Noun+A3sg+P3sg+Loc Noun+A3sg+Pnon+Nom : correct = Verb+Pos^DB+Adj+Agt^DB+Noun+Zero+A3pl+P3sg+Nom
1 : Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Verb+Become+Pos+Fut+A3sg : fors : Noun+Prop+A3sg+Pnon+Gen Noun+A3sg+P3sg+Acc : correct = Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Verb+Become+Pos^DB+Adj+FutPart
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3pl+P3pl+Acc : fors : Conj Verb+Pos^DB+Adj+PresPart : correct = Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3pl+P3sg+Acc
2 : Noun+A3sg+P3pl+Nom : fors : Noun+A3sg+Pnon+Ins Adj : correct = Noun+A3pl+P3sg+Nom
1 : Verb+Pos^DB+Adj+FutPart^DB+Noun+Zero+A3pl+P3pl+Nom : fors : Noun+A3sg+Pnon+Dat Conj : correct = Verb+Pos^DB+Adj+FutPart^DB+Noun+Zero+A3sg+P3pl+Nom
1 : Noun+A3sg+Pnon+Ins : fors : Noun+A3sg+Pnon+Gen Verb+Pos^DB+Noun+Inf3+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Ins
1 : Verb+Pos^DB+Noun+Inf2+A3sg+P3sg+Nom : fors : Noun+A3pl+Pnon^DB+Adverb+Ly Punc : correct = Noun+A3sg+P3sg+Nom
2 : Verb+Pos+Imp+A2sg : pre : Punc Num+Range : correct = Noun+A3sg+Pnon+Nom
2 : Noun+A3pl+P3pl+Dat : fors : Postp Adverb+AdjMdfy : correct = Noun+A3pl+P3sg+Dat
2 : Noun+A3pl+P3pl+Dat : fors : Noun+A3sg+P3sg+Nom Verb+Pos+Past+A3sg : correct = Noun+A3pl+P3sg+Dat
2 : Noun+A3pl+P3sg+Ins : pre : Noun+Prop+A3sg+Pnon+Nom Noun+Prop+A3sg+Pnon+Nom : correct = Noun+A3pl+P3pl+Ins
2 : Noun+A3sg+P3sg+Acc : fors : Noun+A3sg+Pnon+Loc Num+Range : correct = Noun+A3sg+P2sg+Acc
2 : Noun+Prop+A3sg+P3sg+Loc : fors : Pron+Reflex+A1sg+P1sg+Acc Verb+Pos+Prog1+Cond+A1sg : correct = Noun+Prop+A3sg+Pnon+Loc
2 : Noun+A3pl+P3sg+Loc : pre : Noun+Prop+A3sg+P3sg+Loc Noun+Prop+A3sg+Pnon+Nom : correct = Noun+A3pl+P3pl+Loc
2 : Noun+A3pl+P3sg+Loc : fors : Conj Num+Card : correct = Noun+A3pl+P3pl+Loc
1 : Noun+A3pl+P3sg+Loc : fors : Adj Noun+A3sg+Pnon+Acc : correct = Noun+A3pl+P3pl+Loc
1 : Noun+A3pl+P3sg+Loc : fors : Noun+A3sg+Pnon+Loc Verb^DB+Verb+AbleNeg+Neg^DB+Adj+PresPart : correct = Noun+A3pl+P3pl+Loc
1 : Noun+A3pl+P3sg+Loc : fors : Noun+A3sg+Pnon+Nom Noun+A3sg+Pnon+Nom : correct = Noun+A3pl+P3pl+Loc
1 : Noun+A3pl+P3sg+Loc : pre : Noun+Prop+A3sg+Pnon+Nom Noun+A3sg+P3sg+Loc : correct = Noun+A3sg+P3pl+Loc
1 : Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+P3sg+Nom : fors : Noun+A3sg+P3sg+Abl Verb^DB+Verb+Pass+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Acc : correct = Noun+A3sg+P3sg+Nom
1 : Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+P3sg+Nom : fors : Punc Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom^DB+Adj+FitFor : fors : Verb^DB+Verb+Caus^DB+Verb+Pass+Pos+Aor+A3sg^DB+Adverb+While Punc : correct = Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom
1 : Verb+Pos^DB+Adj+PresPart : pre : Adj^DB+Noun+Zero+A3sg+P3sg+Loc Noun+Prop+A3pl+P3sg+Gen : correct = Verb^DB+Verb+Pass+Pos^DB+Adj+PresPart
2 : Noun+A3sg+P3pl+Acc : pre : Noun+A3sg+P3sg+Acc Noun+A3sg+P3sg+Loc : correct = Noun+A3pl+P3sg+Acc

2 : Noun+A3sg+P3pl+Acc : pre : Noun+A3sg+P3sg+Loc Postp : correct = Noun+A3pl+P3sg+Acc
1 : Verb^DB+Verb+Pass+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc : fors :
Noun+A3sg+Pnon+Nom Conj : correct =
Verb^DB+Verb+Pass+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom
1 : Verb^DB+Verb+Pass+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc : fors : Conj
Noun+Prop+A3sg+Pnon+Nom : correct =
Verb^DB+Verb+Pass+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom
1 : Noun+A3sg+Pnon+Nom^DB+Adj+Rel : fors : Conj Verb+Pos^DB+Noun+Inf3+A3sg+Pnon+Nom :
correct = Postp^DB+Adj+Rel
1 : Noun+A3sg+Pnon+Nom^DB+Adj+Rel : fors : Noun+A3sg+Pnon+Nom
Verb^DB+Verb+Pass+Pos^DB+Adj+PresPart : correct = Postp^DB+Adj+Rel
1 : Noun+A3sg+Pnon+Nom^DB+Adj+Rel : fors : Noun+A3sg+Pnon+Nom Noun+Prop+A3sg+Pnon+Loc :
correct = Postp^DB+Adj+Rel
2 : Noun+A3pl+P3sg+Gen : pre : Noun+A3pl+P3sg+Gen Noun+A3sg+Pnon+Nom : correct =
Noun+A3sg+P3pl+Gen
2 : Noun+A3pl+P3pl+Gen : pre : Noun+A3sg+Pnon+Nom Conj : correct = Noun+A3pl+P3sg+Gen
2 : Noun+A3pl+P3pl+Gen : pre : Noun+A3sg+Pnon+Nom Noun+A3sg+P3sg+Nom : correct =
Noun+A3pl+P3sg+Gen
2 : Noun+A3pl+P3pl+Gen : pre : Noun+A3sg+Pnon+Gen Num+Card : correct = Noun+A3pl+P3sg+Gen
2 : Noun+A3pl+P3pl+Acc : pre : Noun+Prop+A3sg+Pnon+Nom Punc : correct = Noun+A3pl+P3sg+Acc
2 : Pron+A1sg+Pnon+Nom : pre : Noun+Prop+A3sg+Pnon+Nom Noun+A3sg+P3sg+Nom : correct =
Noun+Prop+A3sg+Pnon+Nom
1 : Verb+Pos^DB+Adj+FutPart^DB+Noun+Zero+A3sg+P3sg+Nom : fors : Noun+A3sg+Pnon+Nom
Noun+A3sg+P3sg+Nom : correct =
Verb^DB+Verb+Pass+Pos^DB+Adj+FutPart^DB+Noun+Zero+A3sg+Pnon+Acc
1 : Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : fors :
Noun+A3sg+Pnon+Nom Ques+Pres+A3sg : correct =
Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Adj+FitFor
1 : Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : fors :
Verb+Pos^DB+Noun+Inf2+A3sg+P3sg+Nom Conj : correct =
Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Adj+FitFor
1 : Verb^DB+Verb+Pass+Pos^DB+Adj+FutPart : pre : Noun+Prop+A3sg+P3sg+Loc
Noun+A3sg+Pnon+Nom : correct = Verb^DB+Verb+Pass+Pos+Fut+A3sg
1 : Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+P3sg+Nom : fors :
Noun+Prop+A3sg+Pnon+Nom Noun+A3sg+P3sg+Abl : correct =
Num+Card^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+P3sg+Nom
1 : Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+P3sg+Nom : pre : Adj Punc : correct =
Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Acc
1 : Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+P3sg+Nom : fors : Verb+Pos+Past+A3sg
Punc : correct = Noun+A3sg+P3sg+Nom
2 : Interj : pre : Punc Adverb : correct = Noun+A3sg+Pnon+Nom
2 : Interj : fors : Noun+A3sg+P3sg+Nom Noun+A3sg+Pnon+Nom : correct =
Noun+Prop+A3sg+Pnon+Nom
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom : fors : Punc
Noun+Prop+A3sg+Pnon+Nom : correct = Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom : fors : Noun+A3sg+Pnon+Nom
Verb+Pos^DB+Noun+Inf2+A3sg+P3sg+Loc : correct =
Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom : fors : Noun+A3sg+Pnon+Nom
Verb^DB+Verb+Pass+Pos^DB+Noun+Inf3+A3pl+P3pl+Dat : correct =
Verb^DB+Verb+Pass+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom : fors : Punc Noun+A3sg+Pnon+Nom :
correct = Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom : fors : Punc Noun+A3sg+P3sg+Ins :
correct = Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc

1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom : fors : Noun+A3sg+Pnon+Nom
Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+P3sg+Gen : correct =
Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom : pre :
Verb^DB+Verb+Caus+Pos^DB+Adverb+ByDoingSo Noun+A3pl+Pnon+Dat : correct =
Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom : pre : Noun+A3sg+P3sg+Dat
Noun+Prop+A3sg+Pnon+Nom : correct = Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom : pre : Noun+A3sg+P3sg+Gen
Verb+Pos^DB+Noun+Inf2+A3sg+Pnon+Nom : correct =
Verb^DB+Verb+Pass+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom
1 : Noun+A3sg+Pnon+Nom^DB+Adj+Agt : fors : Verb+Pos^DB+Adverb+AfterDoingSo
Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P1pl+Nom : correct =
Noun+A3sg+Pnon+Nom^DB+Noun+Agt+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom^DB+Adj+Agt : fors : Verb+Pos^DB+Adverb+When Noun+A3sg+Pnon+Nom :
correct = Noun+A3sg+Pnon+Nom^DB+Noun+Agt+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom^DB+Adj+Agt : fors : Punc Noun+A3sg+Pnon+Nom : correct =
Noun+A3sg+Pnon+Nom^DB+Noun+Agt+A3sg+Pnon+Nom
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3pl+P3pl+Nom : fors : Noun+A3sg+Pnon+Loc Adj :
correct = Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3pl+P3sg+Nom
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3pl+P3pl+Nom : fors :
Verb+Pos^DB+Noun+Inf2+A3sg+Pnon+Loc Punc : correct =
Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3pl+Pnon+Acc
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3pl+P3pl+Nom : pre :
Noun+A3sg+Pnon+Nom^DB+Adj+Without Conj : correct =
Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3pl+Nom
2 : Noun+A3pl+P3sg+Abl : pre : Adj Noun+A3sg+Pnon+Gen : correct = Noun+A3pl+P3pl+Abl
2 : Conj : pre : Punc Noun+A3sg+Pnon+Acc : correct = Adverb
2 : Conj : pre : Punc Verb+Pos+Narr+A3sg : correct = Adverb
2 : Conj : pre : Punc Noun+A3sg+P1pl+Acc : correct = Adverb
1 : Conj : fors : Noun+A3sg+Pnon+Nom^DB+Noun+Agt+A3pl+Pnon+Dat Noun+A3sg+Pnon+Nom :
correct = Adverb
1 : Conj : fors : Noun+A3sg+Pnon+Abl Adj : correct = Adverb
2 : Noun+A3pl+P3sg+Nom : pre : Noun+A3pl+P3sg+Gen Noun+Prop+A3sg+Pnon+Nom : correct =
Noun+A3pl+P3pl+Nom
2 : Noun+A3pl+P3sg+Nom : fors : Adverb Noun+A3sg+Pnon+Loc : correct = Noun+A3pl+Pnon+Acc
2 : Noun+A3pl+P3sg+Nom : fors : Adj Noun+Prop+A3sg+Pnon+Nom : correct = Noun+A3pl+Pnon+Acc
2 : Noun+A3pl+P3sg+Nom : fors : Noun+A3sg+Pnon+Dat Postp : correct = Noun+A3pl+P3pl+Nom
1 : Noun+A3pl+P3sg+Nom : fors : Postp Adj : correct = Noun+A3pl+P3pl+Nom
1 : Verb^DB+Verb+Pass+Pos+Prog1+A3sg : pre : Conj Noun+A3pl+P3sg+Nom : correct =
Verb+Pos+Prog1+A3sg
1 : Verb^DB+Verb+Pass+Pos+Prog1+A3sg : pre : Verb+Pos^DB+Adj+Zero Noun+A3sg+Pnon+Dat :
correct = Verb+Pos+Prog1+A3sg
2 : Verb+Past+A3sg : pre : Noun+A3sg+P3sg+Dat Punc : correct = Verb+Pos+Past+A3sg
2 : Verb+Pos+Fut+A3sg : fors : Noun+A3sg+Pnon+Gen Noun+A3sg+P3sg+Gen : correct = Adj
2 : Verb+Pos+Fut+A3sg : fors : Adj Noun+A3sg+Pnon+Nom : correct = Adj
2 : Verb+Pos+Fut+A3sg : fors : Noun+A3pl+Pnon+Nom Verb+Pos+Aor+A3sg : correct = Adj
1 : Verb+Pos+Fut+A3sg : fors : Noun+Prop+A3sg+Pnon+Nom Punc : correct = Verb+Pos^DB+Adj+FutPart
1 : Adj^DB+Noun+Zero+A3sg+P3pl+Loc^DB+Adj+Rel : fors : Noun+Prop+A3sg+Pnon+Nom
Noun+A3sg+P3sg+Nom : correct = Adj^DB+Noun+Zero+A3pl+P3sg+Loc^DB+Adj+Rel
1 : Adj^DB+Noun+Zero+A3pl+P3pl+Loc : fors : Punc Noun+Prop+A3sg+Pnon+Nom : correct =
Adj^DB+Noun+Zero+A3sg+P3pl+Loc
1 : Adj^DB+Noun+Zero+A3pl+P3pl+Loc : fors : Noun+A3sg+Pnon+Nom Punc : correct =
Adj^DB+Noun+Zero+A3sg+P3pl+Loc
1 : Verb+Pos+Past+A3sg : pre : Noun+A3sg+P3sg+Nom Verb^DB+Verb+Pass+Pos^DB+Adj+Zero : correct
= Verb^DB+Verb+Pass+Pos+Past+A3sg

1 : Verb+Pos+Past+A3sg : pre : Noun+A3pl+P3pl+Nom Verb^DB+Verb+Pass+Pos^DB+Adj+Zero : correct = Verb^DB+Verb+Pass+Pos+Past+A3sg
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc : fors : Verb^DB+Verb+Caus^DB+Verb+Pass+Pos+Prog1+A3sg Punc : correct = Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc : pre : Noun+A3sg+P3pl+Gen Noun+A3pl+P3sg+Gen : correct = Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom
2 : Noun+A3sg+P3pl+Gen : pre : Punc Noun+A3sg+P3sg+Nom : correct = Noun+A3pl+P3sg+Gen
2 : Adverb+AdjMdfy : pre : Noun+A3pl+P3sg+Gen Noun+A3sg+Pnon+Nom : correct = Adj
2 : Adverb+AdjMdfy : pre : Adverb Noun+A3sg+P1pl+Nom : correct = Adj
2 : Adverb+AdjMdfy : pre : Pron+A1sg+Pnon+Abl Noun+A3sg+Pnon+Nom : correct = Adj
2 : Adverb+AdjMdfy : fors : Noun+A3sg+Pnon+Nom Conj : correct = Adj
2 : Adverb+AdjMdfy : fors : Noun+A3sg+Pnon+Gen Noun+A3pl+P3sg+Gen : correct = Adj
2 : Adverb+AdjMdfy : fors : Noun+A3sg+Pnon+Dat Conj : correct = Adj
1 : Adverb+AdjMdfy : fors : Noun+A3sg+Pnon+Nom Verb+Pos^DB+Adj+PresPart : correct = Adj
1 : Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : fors : Noun+A3sg+P3sg+Nom Conj : correct = Noun+A3sg+Pnon+Nom^DB+Adj+FitFor
1 : Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : fors : Noun+A3sg+P3sg+Nom Noun+A3pl+P3sg+Gen : correct = Noun+A3sg+Pnon+Nom^DB+Adj+FitFor
1 : Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : fors : Noun+A3sg+P3sg+Nom Noun+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Nom^DB+Adj+FitFor
1 : Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : fors : Noun+A3sg+P3sg+Acc Verb^DB+Verb+Caus+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Abl : correct = Noun+A3sg+Pnon+Nom^DB+Adj+FitFor
1 : Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : fors : Noun+A3sg+P3sg+Loc Verb+Pos^DB+Noun+Inf1+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Nom^DB+Adj+FitFor
1 : Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : fors : Noun+A3pl+P3sg+Nom Noun+A3sg+Pnon+Dat : correct = Noun+A3sg+Pnon+Nom^DB+Adj+FitFor
1 : Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : fors : Noun+A3pl+P3sg+Nom Punc : correct = Noun+A3sg+Pnon+Nom^DB+Adj+FitFor
1 : Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : fors : Noun+A3pl+P3pl+Nom Postp : correct = Noun+A3sg+Pnon+Nom^DB+Adj+With^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Adj+FitFor
1 : Verb+Pos^DB+Adj+Agt : fors : Noun+A3sg+P3sg+Gen Punc : correct = Noun+A3sg+Pnon+Nom
2 : Noun+A3sg+Pnon+Dat : pre : Noun+A3sg+Pnon+Nom Noun+A3pl+Pnon+Acc : correct = Noun+A3sg+Pnon+Nom
2 : Noun+A3sg+Pnon+Dat : pre : Noun+A3sg+Pnon+Nom Verb+Neg+Prog1+Cond+A3sg : correct = Noun+A3sg+P3sg+Dat
2 : Noun+A3sg+Pnon+Dat : pre : Noun+A3sg+Pnon+Nom Adverb : correct = Noun+A3sg+P3sg+Dat
2 : Noun+A3sg+Pnon+Dat : fors : Conj Noun+A3sg+P3sg+Acc : correct = Noun+A3sg+P3sg+Dat
1 : Noun+A3sg+Pnon+Dat : fors : Noun+A3sg+P3sg+Loc Punc : correct = Noun+A3sg+Pnon+Nom
2 : Noun+A3pl+P3pl+Nom : fors : Noun+A3sg+P3sg+Acc Verb+Pos+Past+A3sg : correct = Noun+A3pl+Pnon+Acc
1 : Noun+A3pl+P3pl+Nom : fors : Noun+A3sg+P3sg+Loc Verb^DB+Verb+Pass+Pos^DB+Noun+Inf2+A3sg+P3sg+Nom : correct = Noun+A3pl+P3sg+Nom
1 : Noun+A3sg+P3sg+Loc : fors : Verb+Pos^DB+Adverb+AfterDoingSo Adj : correct = Adverb
2 : Adverb : pre : Noun+A3pl+P3pl+Ins Punc : correct = Postp
2 : Adverb : pre : Punc Noun+A3sg+Pnon+Dat : correct = Noun+A3sg+Pnon+Nom
2 : Adverb : pre : Noun+A3pl+P3pl+Acc Postp : correct = Noun+A3sg+Pnon+Nom
2 : Adverb : pre : Noun+A3pl+Pnon+Acc Conj : correct = Adj
2 : Adverb : pre : Adj Noun+A3sg+P3sg+Gen : correct = Noun+A3sg+Pnon+Nom
2 : Adverb : pre : Noun+A3sg+Pnon+Ins Noun+A3sg+P3sg+Nom : correct = Postp
2 : Adverb : pre : Noun+Prop+A3sg+Pnon+Ins Noun+Prop+A3sg+Pnon+Nom : correct = Postp
2 : Adverb : pre : Noun+A3sg+P3sg+Ins Adj : correct = Postp
2 : Adverb : pre : Noun+A3pl+Pnon+Ins Noun+A3pl+Pnon+Gen : correct = Postp
2 : Adverb : pre : Verb+Neg+Desr+A3sg Postp : correct = Conj
2 : Adverb : fors : Noun+Prop+A3sg+Pnon+Gen Noun+A3pl+P3sg+Acc : correct = Postp
2 : Adverb : fors : Noun+Prop+A3sg+Pnon+Nom Conj : correct = Postp

2 : Adverb : fors : Noun+A3sg+Pnon+Abl Noun+A3sg+P1pl+Dat : correct = Noun+A3sg+P3pl+Nom
2 : Adverb : fors : Noun+A3pl+Pnon+Nom Noun+Prop+A3sg+Pnon+Loc : correct = Conj
1 : Adverb : fors : Adj Noun+A3pl+Pnon+Acc : correct =
Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Ins
1 : Adverb : pre : Noun+A3sg+Pnon+Nom^DB+Noun+Agt+A3sg+Pnon+Ins Noun+A3sg+Pnon+Nom :
correct = Postp
1 : Adverb : pre : Verb+Pos^DB+Noun+Inf2+A3sg+P3sg+Ins Noun+A3pl+P3sg+Gen : correct = Postp
1 : Noun+A3sg+Pnon+Nom^DB+Adj+With : fors :
Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom Noun+A3pl+P3sg+Nom : correct = Adverb
1 : Noun+A3sg+Pnon+Nom^DB+Adj+With : fors : Verb+Pos^DB+Adj+PresPart
Noun+Prop+A3sg+Pnon+Gen : correct = Noun+A3sg+Pnon+Nom
2 : Noun+A3sg+Pnon+Acc : pre : Noun+A3sg+Pnon+Nom Noun+A3pl+Pnon+Acc : correct =
Noun+A3sg+P3sg+Nom
2 : Noun+A3sg+Pnon+Acc : pre : Noun+Prop+A3sg+Pnon+Gen Noun+A3sg+P3sg+Nom : correct =
Noun+A3sg+P3sg+Nom
2 : Noun+A3sg+Pnon+Acc : pre : Adj Noun+A3sg+Pnon+Gen : correct = Noun+A3sg+P3sg+Nom
2 : Noun+A3sg+Pnon+Acc : pre : Postp Noun+A3pl+P3pl+Dat : correct = Noun+A3sg+P3sg+Nom
2 : Noun+A3sg+Pnon+Acc : fors : Adverb Noun+A3sg+Pnon+Loc : correct = Noun+A3sg+P3sg+Nom
2 : Noun+A3sg+Pnon+Acc : fors : Verb+Pos+Narr+Past+A3sg Punc : correct = Noun+A3sg+P3sg+Nom
2 : Noun+A3sg+Pnon+Acc : fors : Verb+Pos+Fut+A3sg Punc : correct = Noun+A3sg+P3sg+Nom
2 : Noun+A3sg+Pnon+Acc : fors : Noun+A3sg+P3sg+Nom Adj : correct = Noun+A3sg+P3sg+Nom
2 : Noun+A3sg+Pnon+Acc : fors : Adj Adj : correct = Noun+A3sg+P3sg+Nom
2 : Noun+A3sg+Pnon+Acc : fors : Noun+A3sg+Pnon+Nom Noun+A3sg+P3sg+Ins : correct =
Noun+A3sg+P3sg+Nom
2 : Noun+A3sg+Pnon+Acc : fors : Conj Adj : correct = Noun+A3sg+P3sg+Nom
1 : Noun+A3sg+Pnon+Acc : fors : Noun+A3pl+Pnon+Nom
Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Ins : correct =
Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Adj+Agt
1 : Noun+A3sg+Pnon+Acc : fors : Verb+Pos+Aor+A3sg^DB+Adverb+While Punc : correct =
Noun+A3sg+P3sg+Nom
1 : Noun+A3sg+Pnon+Acc : fors : Noun+A3sg+P3sg+Nom Noun+A3sg+Pnon+Nom : correct =
Noun+A3sg+P3sg+Nom
2 : Noun+A3sg+Pnon+Nom : pre : Verb+Pos+Aor+A3sg Noun+A3sg+Pnon+Ins : correct = Adj
2 : Noun+A3sg+Pnon+Nom : pre : Verb+Neg+Past+A2pl Adverb+Ques : correct =
Noun+Prop+A3sg+Pnon+Nom
2 : Noun+A3sg+Pnon+Nom : pre : Adverb Noun+A3sg+P3sg+Acc : correct =
Noun+Prop+A3sg+Pnon+Nom
2 : Noun+A3sg+Pnon+Nom : pre : Adverb+AdjMdfy Noun+Prop+A3sg+Pnon+Gen : correct = Adj
2 : Noun+A3sg+Pnon+Nom : pre : Noun+A3sg+P2pl+Nom Pron+A2pl+Pnon+Gen : correct =
Verb+Pos+Imp+A2sg
2 : Noun+A3sg+Pnon+Nom : fors : Noun+Prop+A3sg+Pnon+Nom Interj : correct =
Noun+Prop+A3sg+Pnon+Nom
2 : Noun+A3sg+Pnon+Nom : fors : Noun+Prop+A3sg+Pnon+Dat Punc : correct =
Noun+Prop+A3sg+Pnon+Nom
2 : Noun+A3sg+Pnon+Nom : fors : Postp Noun+A3sg+Pnon+Gen : correct =
Noun+Prop+A3sg+Pnon+Nom
2 : Noun+A3sg+Pnon+Nom : fors : Noun+Prop+A3sg+Pnon+Dat Verb+Pos+Past+A3sg : correct =
Noun+A3sg+Pnon+Dat
2 : Noun+A3sg+Pnon+Nom : fors : Noun+Prop+A3sg+Pnon+Loc Verb+Pos+Prog1+A3sg : correct =
Noun+Prop+A3sg+Pnon+Nom
2 : Noun+A3sg+Pnon+Nom : fors : Noun+A3pl+Pnon+Loc Noun+A3sg+Pnon+Gen : correct = Adj
2 : Noun+A3sg+Pnon+Nom : fors : Noun+A3pl+P3sg+Acc Verb+Pos+Narr+Past+A3sg : correct =
Noun+Prop+A3sg+Pnon+Nom
2 : Noun+A3sg+Pnon+Nom : fors : Noun+A3sg+Pnon+Loc Num+Card : correct = Conj
2 : Noun+A3sg+Pnon+Nom : fors : Noun+Prop+A3sg+Pnon+Ins Verb+Pos+Past+A3sg : correct =
Noun+Prop+A3sg+Pnon+Nom

2 : Noun+A3sg+Pnon+Nom : fors : Noun+A3pl+Pnon+Acc Adverb : correct = Noun+Prop+A3sg+Pnon+Nom
2 : Noun+A3sg+Pnon+Nom : fors : Noun+A3sg+Pnon+Acc Verb+Pos+Narr+A3sg : correct = Noun+Prop+A3sg+Pnon+Nom
2 : Noun+A3sg+Pnon+Nom : fors : Postp Verb+Pos+Past+A1sg : correct = Noun+Prop+A3sg+Pnon+Nom
2 : Noun+A3sg+Pnon+Nom : pre : Noun+Prop+A3sg+P2sg+Loc : correct = Noun+Prop+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom : fors : Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom Noun+A3sg+P3sg+Nom : correct = Noun+Prop+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom : fors : Noun+A3sg+Pnon+Dat Verb^DB+Verb+Pass+Pos+Neces+Cop+A3sg : correct = Noun+A3sg+P2sg+Nom^DB+Adj+With
1 : Noun+A3sg+Pnon+Nom : fors : Noun+A3sg+P3sg+Nom Noun+A3pl+P3sg+Abl : correct = Num+Card^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom : fors : Noun+A3sg+P3sg+Nom Adverb : correct = Noun+Prop+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom : fors : Noun+A3sg+Pnon+Nom Verb+Pos+Prog1+A3sg : correct = Noun+A3sg+Pnon+Nom^DB+Adj+Agt
1 : Noun+A3sg+Pnon+Nom : fors : Noun+A3pl+P3sg+Nom Verb^DB+Verb+Pass+Pos^DB+Adj+PresPart : correct = Noun+Prop+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom : fors : Noun+Prop+A3sg+P3sg+Gen Verb+Pos^DB+Noun+Inf2+A3sg+Pnon+Nom : correct = Noun+Prop+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom : fors : Noun+A3sg+P3sg+Nom Verb^DB+Verb+Pass+Pos^DB+Adj+FutPart : correct = Noun+Prop+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom : fors : Punc Verb+Pos^DB+Adj+Agt^DB+Noun+Zero+A3sg+Pnon+Abl : correct = Noun+Prop+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom : fors : Noun+A3sg+Pnon+Dat Verb^DB+Verb+Caus^DB+Verb+Pass+Pos+Aor+A3sg^DB+Adverb+While : correct = Noun+Prop+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom : fors : Noun+A3pl+Pnon+Nom Verb^DB+Verb+Pass+Pos+Fut+A3sg : correct = Noun+A3sg+Pnon+Nom^DB+Adj+With
1 : Noun+A3sg+Pnon+Nom : fors : Noun+A3pl+Pnon+Acc Noun+A3sg+P3sg+Nom : correct = Adj
1 : Noun+A3sg+Pnon+Nom : fors : Noun+A3pl+Pnon+Loc^DB+Adj+Rel Noun+A3pl+Pnon+Nom : correct = Adj
1 : Noun+A3sg+Pnon+Nom : fors : Noun+A3sg+P3sg+Nom Num+Real^DB+Noun+Zero+A3sg+Pnon+Loc : correct = Noun+Prop+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom : pre : Postp Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : correct = Conj
1 : Noun+A3sg+Pnon+Nom : pre : Verb^DB+Verb+Caus+Pos^DB+Adj+PresPart Adj : correct = Noun+Prop+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom : pre : Punc Verb+Pos^DB+Noun+Inf2+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Nom^DB+Adj+Agt
1 : Noun+A3sg+Pnon+Nom : pre : Noun+Prop+A3sg+Pnon+Nom Noun+A3pl+Pnon+Gen : correct = Noun+Prop+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom : pre : Verb+Pos^DB+Noun+Inf1+A3sg+Pnon+Nom Noun+A3sg+P3sg+Acc : correct = Adverb
1 : Noun+A3sg+Pnon+Nom : pre : Punc Noun+A3sg+Pnon+Nom^DB+Adj+With^DB+Noun+Zero+A3pl+Pnon+Nom : correct = Noun+Prop+A3sg+Pnon+Nom
1 : Noun+A3sg+Pnon+Nom : pre : Verb+Pos^DB+Noun+Inf2+A3sg+Pnon+Dat Punc : correct = Verb+Pos+Aor+A3sg
2 : Adverb+Ques : pre : Noun+A3sg+Pnon+Nom Verb+Pres+Cop+A3sg : correct = Adj+Ques
2 : Adverb+Ques : pre : Noun+A3sg+P3sg+Ins Noun+A3sg+Pnon+Nom : correct = Pron+Ques+A3sg+Pnon+Nom
2 : Adverb+Ques : pre : Postp Noun+A3sg+P3sg+Nom : correct = Conj
2 : Adverb+Ques : pre : Verb+Pos+Past+A3sg Noun+Prop+A3sg+Pnon+Loc : correct = Conj
2 : Adverb+Ques : pre : Noun+A3sg+Pnon+Nom Noun+A3sg+Pnon+Nom : correct = Pron+Ques+A3sg+Pnon+Nom
2 : Adverb+Ques : pre : Punc Postp : correct = Adj+Ques

2 : Adverb+Ques : pre : Noun+A3sg+Pnon+Gen Adj : correct = Adj+Ques
2 : Adverb+Ques : pre : Noun+A3pl+P3sg+Loc Noun+A3sg+Pnon+Nom : correct = Adj+Ques
2 : Adverb+Ques : pre : Noun+A3sg+Pnon+Loc Noun+A3sg+Pnon+Nom : correct = Pron+Ques+A3sg+Pnon+Nom
2 : Adverb+Ques : pre : Noun+A3sg+P3sg+Loc Noun+A3pl+Pnon+Nom : correct = Pron+Ques+A3sg+Pnon+Nom
2 : Adverb+Ques : pre : Noun+Prop+A3sg+Pnon+Loc : correct = Adj+Ques
2 : Noun+A3pl+Pnon+Acc : pre : Noun+A3sg+Pnon+Gen Adj : correct = Noun+A3pl+P3sg+Nom
2 : Noun+A3pl+Pnon+Acc : pre : Noun+A3sg+Pnon+Nom Noun+A3sg+Pnon+Dat : correct = Noun+A3pl+P3sg+Nom
2 : Noun+A3pl+Pnon+Acc : pre : Adj Pron+A3sg+Pnon+Gen : correct = Noun+A3pl+P3sg+Nom
2 : Noun+A3pl+Pnon+Acc : pre : Conj Noun+A3pl+Pnon+Nom : correct = Noun+A3pl+P3sg+Nom
2 : Noun+A3pl+Pnon+Acc : pre : Noun+Prop+A3sg+Pnon+Nom Punc : correct = Noun+A3pl+P3sg+Nom
2 : Noun+A3pl+Pnon+Acc : fors : Noun+A3pl+Pnon+Gen Noun+A3sg+P3sg+Nom : correct = Noun+A3pl+P3sg+Nom
2 : Noun+A3pl+Pnon+Acc : fors : Punc Noun+A3sg+P3sg+Nom : correct = Noun+A3pl+P3sg+Nom
2 : Noun+Prop+A3sg+Pnon+Nom : pre : Noun+A3pl+Pnon+Acc Noun+A3sg+Pnon+Gen : correct = Noun+A3sg+Pnon+Nom
2 : Noun+Prop+A3sg+Pnon+Nom : pre : Noun+A3sg+Pnon+Abl Adj : correct = Adj
2 : Noun+Prop+A3sg+Pnon+Nom : pre : Noun+A3sg+Pnon+Nom Noun+A3sg+P3sg+Gen : correct = Noun+A3sg+P3sg+Nom
2 : Noun+Prop+A3sg+Pnon+Nom : pre : Noun+A3sg+Pnon+Abl Verb+Pos+Desr+A1pl : correct = Noun+A3sg+Pnon+Nom
2 : Noun+Prop+A3sg+Pnon+Nom : fors : Postp Noun+A3sg+P3sg+Ins : correct = Noun+A3sg+Pnon+Nom
2 : Noun+Prop+A3sg+Pnon+Nom : fors : Noun+A3sg+P3sg+Ins Noun+Prop+A3sg+Pnon+Dat : correct = Noun+A3sg+Pnon+Nom
2 : Noun+Prop+A3sg+Pnon+Nom : fors : Noun+Prop+A3sg+P3sg+Dat Noun+Prop+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Nom
2 : Noun+Prop+A3sg+Pnon+Nom : fors : Noun+A3pl+P3pl+Loc Conj : correct = Noun+A3sg+Pnon+Nom
1 : Noun+Prop+A3sg+Pnon+Nom : fors : Noun+A3sg+P3sg+Ins Adj : correct = Noun+A3sg+Pnon+Nom
1 : Noun+Prop+A3sg+Pnon+Nom : fors : Verb^DB+Verb+Pass+Pos^DB+Noun+Inf3+A3sg+P3sg+Abl Adj : correct = Noun+A3sg+P3sg+Nom
1 : Noun+Prop+A3sg+Pnon+Nom : fors : Postp Adj^DB+Noun+Zero+A3sg+Pnon^DB+Adverb+Ly : correct = Noun+A3sg+Pnon+Nom
1 : Noun+Prop+A3sg+Pnon+Nom : pre : Adverb+AdjMdfy Adverb+AdjMdfy : correct = Noun+A3sg+Pnon+Nom^DB+Adj+With
1 : Noun+Prop+A3sg+Pnon+Nom : pre : Conj Verb^DB+Verb+Recip+Pos^DB+Noun+Inf2+A3sg+Pnon+Loc : correct = Verb+Pos^DB+Adj+PresPart
1 : Noun+Prop+A3sg+Pnon+Nom : pre : Postp Verb^DB+Verb+Pass+Pos^DB+Noun+Inf2+A3sg+P3sg+Nom : correct = Noun+A3sg+Pnon+Nom
2 : Adj : pre : Num+Card Noun+A3sg+P3sg+Loc : correct = Noun+A3sg+Pnon+Nom
2 : Adj : pre : Verb+Pos+Aor+A3sg Noun+A3sg+P3sg+Dat : correct = Pron+A3sg+Pnon+Nom
2 : Adj : pre : Pron+A3pl+Pnon+Ins Noun+Prop+A3sg+Pnon+Nom : correct = Adverb
2 : Adj : pre : Verb+Pos+Desr+A2sg Noun+A3pl+Pnon+Nom : correct = Adverb
2 : Adj : pre : Noun+A3pl+Pnon+Gen Noun+A3sg+P3sg+Loc : correct = Noun+A3sg+Pnon+Nom
2 : Adj : pre : Noun+Prop+A3sg+Pnon+Loc Conj : correct = Noun+A3sg+Pnon+Nom
2 : Adj : pre : Postp Noun+A3sg+P1sg+Nom : correct = Adverb
2 : Adj : pre : Conj Noun+A3pl+Pnon+Dat : correct = Noun+A3sg+Pnon+Nom
2 : Adj : pre : Noun+A3sg+P1sg+Nom Noun+A3sg+P1sg+Ins : correct = Adverb
2 : Adj : pre : Noun+A3sg+P3sg+Ins Noun+A3pl+Pnon+Gen : correct = Noun+A3sg+Pnon+Nom
2 : Adj : pre : Adj Noun+A3sg+P1pl+Nom : correct = Noun+A3sg+Pnon+Nom
2 : Adj : pre : Pron+A3sg+Pnon+Gen Noun+A3sg+P3sg+Nom : correct = Noun+Prop+A3sg+Pnon+Nom
2 : Adj : pre : Noun+A3pl+Pnon+Dat Postp : correct = Noun+A3sg+Pnon+Nom
2 : Adj : pre : Pron+Abl Noun+A3sg+P2pl+Nom : correct = Adverb
2 : Adj : pre : Noun+A3sg+P3sg+Ins Noun+A3pl+P3pl+Acc : correct = Noun+A3sg+Pnon+Nom
2 : Adj : pre : Adverb+AdjMdfy Noun+A3sg+P3sg+Acc : correct = Adverb
2 : Adj : fors : Punc Noun+A3sg+P3sg+Loc : correct = Noun+A3sg+Pnon+Nom

2 : Adj : fors : Noun+A3sg+P3sg+Acc Verb+Pos+Past+A3pl : correct = Noun+A3sg+Pnon+Nom
2 : Adj : fors : Adverb Verb+Pos+Past+A3sg : correct = Pron+A3sg+Pnon+Nom
2 : Adj : fors : Verb+Pres+Cop+A3sg Conj : correct = Pron+A3sg+Pnon+Nom
2 : Adj : fors : Noun+A3pl+P1pl+Nom Noun+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Nom
2 : Adj : fors : Conj Noun+A3sg+Pnon+Dat : correct = Pron+A3sg+Pnon+Nom
2 : Adj : fors : Postp Noun+A3pl+P3pl+Acc : correct = Noun+A3sg+Pnon+Nom
2 : Adj : fors : Noun+A3sg+P3sg+Acc Verb+Pos+Prog1+A3sg : correct = Noun+A3sg+Pnon+Nom
2 : Adj : fors : Verb+Pos+Past+A3sg Num+Range : correct = Noun+A3sg+Pnon+Nom
2 : Adj : fors : Conj Pron+A3pl+Pnon+Abl : correct = Pron+A3sg+Pnon+Nom
2 : Adj : fors : Conj Adverb : correct = Pron+A3sg+Pnon+Nom
2 : Adj : fors : Verb+Pos+Prog1+A1sg Punc : correct = Adverb
2 : Adj : fors : Verb+Neg+Past+A1sg Punc : correct = Adverb+AdjMdfy
2 : Adj : fors : Noun+A3sg+Pnon+Nom Noun+Prop+A3sg+P3sg+Acc : correct = Noun+A3sg+Pnon+Nom
2 : Adj : fors : Verb+Pos+Prog1+A2pl Conj : correct = Adverb
1 : Adj : fors : Verb^DB+Verb+Caus+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom Punc : correct = Noun+A3sg+Pnon+Nom
1 : Adj : fors : Verb+Pos^DB+Adverb+ByDoingSo Noun+A3sg+Pnon+Nom : correct = Adverb
1 : Adj : fors : Punc Noun+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Nom
1 : Adj : fors : Verb^DB+Verb+Pass+Pos+Aor+A3sg^DB+Adverb+While Adj : correct = Pron+A3sg+Pnon+Nom
1 : Adj : fors : Num+Card Noun+Prop+A3sg+Pnon+Nom : correct = Noun+A3sg+P3sg+Nom
1 : Adj : fors : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom Verb^DB+Verb+Pass+Pos+Past+A3sg : correct = Noun+A3sg+Pnon+Nom
1 : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom : pre : Noun+A3sg+Pnon+Nom Noun+A3sg+P3sg+Gen : correct = Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc
1 : Adj : fors : Verb^DB+Verb+Pass+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Acc Verb+Pos+Past+A3sg : correct = Noun+A3sg+Pnon+Nom
1 : Adj : fors : Verb^DB+Verb+Pass+Pos^DB+Adj+FutPart^DB+Noun+Zero+A3sg+P3sg+Nom Verb^DB+Verb+Pass+Pos^DB+Adj+PresPart : correct = Noun+A3sg+Pnon+Nom
1 : Adj : fors : Punc Noun+Prop+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Nom
1 : Adj : fors : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Acc Verb+Pos^DB+Adj+PresPart : correct = Adverb
1 : Adj : fors : Verb^DB+Verb+Pass+Neg^DB+Noun+Inf2+A3sg+P3sg+Nom Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Acc : correct = Adverb+AdjMdfy
1 : Adj : fors : Noun+A3sg+Pnon+Nom Noun+A3sg+Pnon+Nom^DB+Noun+Prop+Ness+A3sg+P3sg+Loc^DB+Adj+Rel : correct = Noun+A3sg+Pnon+Nom
1 : Adj : fors : Punc Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : correct = Pron+A3sg+Pnon+Nom
1 : Adj : pre : Adverb+AdjMdfy Adverb+AdjMdfy : correct = Adverb
1 : Adj : pre : Adverb Noun+A3sg+Pnon+Loc : correct = Adverb
1 : Adj : pre : Adj^DB+Noun+Zero+A3sg+Pnon+Acc Verb^DB+Verb+Pass+Pos^DB+Adj+PresPart : correct = Adverb
1 : Adj : pre : Noun+A3sg+P3sg+Nom Noun+A3sg+Pnon+Gen : correct = Pron+A3sg+Pnon+Nom
1 : Adj : pre : Adverb+AdjMdfy Noun+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Nom
1 : Adj : pre : Noun+A3sg+Pnon+Abl Adverb : correct = Postp
2 : Postp : pre : Noun+Prop+A3sg+Pnon+Nom Conj : correct = Conj
2 : Postp : pre : Noun+A3sg+Pnon+Nom Adj+Ques : correct = Noun+A3sg+Pnon+Nom
2 : Postp : pre : Num+Card Noun+A3pl+P3sg+Loc : correct = Adverb+AdjMdfy
2 : Postp : pre : Num+Card Noun+A3pl+Pnon+Loc : correct = Adverb+AdjMdfy
2 : Postp : pre : Noun+A3sg+P3sg+Acc Noun+A3pl+P3pl+Abl : correct = Noun+A3sg+Pnon+Nom
2 : Postp : pre : Num+Card Num+Card : correct = Adverb+AdjMdfy
2 : Postp : pre : Num+Card Noun+A3sg+Pnon+Nom : correct = Conj
2 : Postp : pre : Noun+Prop+A3sg+Pnon+Nom Noun+A3sg+P3sg+Nom : correct = Conj
1 : Postp : fors : Verb+Pos^DB+Noun+Inf2+A3sg+Pnon+Nom : correct = Adverb
2 : Postp : pre : Num+Ratio Noun+Prop+A3sg+Pnon+Loc : correct = Conj

2 : Postp : pre : Noun+Prop+A3sg+Pnon+Nom Noun+A3pl+P3sg+Nom : correct = Conj
2 : Postp : fors : Verb+Pos+Past+A3sg Noun+A3sg+P1sg+Nom : correct = Adverb
2 : Postp : fors : Noun+Prop+A3sg+Pnon+Nom Adj : correct = Conj
2 : Postp : fors : Noun+A3sg+Pnon+Loc Noun+A3pl+P3sg+Nom : correct = Conj
2 : Postp : fors : Noun+Prop+A3sg+Pnon+Nom Noun+Prop+A3sg+P3sg+Abl : correct = Conj
2 : Postp : fors : Noun+A3pl+P3sg+Loc Adverb+AdjMdfy : correct = Conj
2 : Postp : fors : Noun+A3sg+P3sg+Nom Noun+A3sg+Pnon+Gen : correct = Conj
2 : Postp : fors : Noun+A3sg+Pnon+Dat Adverb : correct = Conj
2 : Postp : fors : Noun+Prop+A3sg+Pnon+Loc Adj : correct = Conj
2 : Postp : fors : Noun+A3pl+Pnon+Acc Noun+A3sg+Pnon+Loc : correct = Conj
2 : Postp : fors : Noun+Prop+A3sg+Pnon+Loc Noun+A3pl+P3pl+Nom : correct = Conj
2 : Postp : fors : Pron+A3pl+Pnon+Gen Noun+A3pl+P3pl+Nom : correct = Conj
2 : Postp : fors : Noun+A3sg+Pnon+Dat Verb+Pos+Prog1+A3sg : correct = Adverb
2 : Postp : fors : Noun+A3sg+Pnon+Nom Punc : correct = Conj
1 : Postp : fors : Noun+A3sg+Pnon+Nom Noun+A3sg+Pnon+Nom^DB+Adj+FitFor : correct = Conj
1 : Postp : fors : Noun+A3sg+Pnon+Nom^DB+Verb+Become+Pos^DB+Noun+Inf2+A3sg+Pnon+Nom Verb+Pos^DB+Adj+PresPart : correct = Conj
1 : Postp : fors : Noun+A3sg+Pnon+Nom Noun+Prop+A3sg+Pnon+Nom : correct = Adverb
1 : Postp : fors : Noun+A3sg+P3sg+Acc Verb+Pos^DB+Adj+PresPart : correct = Adverb
1 : Postp : fors : Adj^DB+Noun+Zero+A3sg+P3pl+Loc Noun+A3sg+Pnon+Nom : correct = Adverb
1 : Postp : fors : Noun+A3sg+Pnon+Nom^DB+Adj+With Verb+Pos^DB+Noun+Inf2+A3pl+Pnon+Gen : correct = Conj
1 : Postp : fors : Noun+A3pl+Pnon+Dat Noun+A3sg+Pnon+Nom : correct = Conj
1 : Postp : fors : Noun+A3sg+Pnon+Nom Noun+A3sg+P3sg+Loc : correct = Conj
1 : Postp : fors : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Verb+Become+Pos^DB+Noun+Inf2+A3sg+Pnon+Nom : correct = Conj
1 : Postp : fors : Noun+Prop+A3sg+Pnon+Loc^DB+Adj+Rel Noun+A3sg+Pnon+Nom : correct = Conj
1 : Postp : fors : Conj Noun+A3sg+P3sg+Loc^DB+Adj+Rel : correct = Adverb
1 : Postp : fors : Noun+A3sg+Pnon+Ins Noun+A3sg+Pnon+Acc : correct = Adverb
1 : Postp : fors : Verb+Pos^DB+Noun+Inf3+A3sg+Pnon+Nom Verb+Pos^DB+Noun+Inf3+A3sg+Pnon+Nom : correct = Conj
1 : Postp : fors : Noun+Prop+A3sg+Pnon+Nom Noun+A3sg+Pnon+Nom^DB+Verb+Become+Pos^DB+Noun+Inf2+A3sg+Pnon+Nom : correct = Adverb
1 : Postp : fors : Verb+Pos^DB+Noun+Inf2+A3sg+P3sg+Nom Noun+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Nom
1 : Postp : fors : Noun+Prop+A3sg+Pnon+Nom Noun+Prop+A3sg+Pnon+Gen : correct = Adverb
1 : Postp : fors : Noun+Prop+A3sg+Pnon+Gen Noun+A3sg+P3sg+Nom : correct = Adverb
1 : Postp : fors : Noun+Prop+A3sg+Pnon+Nom^DB+Adj+With Noun+Prop+A3sg+Pnon+Nom : correct = Conj
1 : Postp : fors : Verb+Pos+Past+A3sg Noun+A3pl+Pnon+Nom : correct = Adverb
1 : Postp : pre : Adverb+AdjMdfy Postp : correct = Adverb
1 : Postp : pre : Noun+A3sg+Pnon+Nom Verb^DB+Verb+Caus^DB+Verb+Pass+Pos^DB+Adverb+When : correct = Conj
1 : Postp : pre : Noun+Prop+A3sg+Pnon+Nom Adj^DB+Noun+Zero+A3sg+Pnon+Nom^DB+Noun+Ness+A3sg+P3sg+Loc : correct = Conj
1 : Postp : pre : Punc Postp : correct = Adverb
2 : Noun+A3sg+P3sg+Nom : pre : Noun+A3sg+P1pl+Nom Noun+Prop+A3sg+Pnon+Dat : correct = Noun+A3sg+Pnon+Acc
2 : Noun+A3sg+P3sg+Nom : pre : Num+Card Noun+A3pl+P3pl+Nom : correct = Noun+A3sg+Pnon+Acc
2 : Noun+A3sg+P3sg+Nom : pre : Noun+A3pl+Pnon+Ins Verb+Pres+A1sg : correct = Noun+A3sg+Pnon+Acc
2 : Noun+A3sg+P3sg+Nom : pre : Adj Noun+A3sg+P3sg+Ins : correct = Noun+A3sg+Pnon+Acc
2 : Noun+A3sg+P3sg+Nom : pre : Noun+A3pl+Pnon+Loc Noun+A3sg+P3sg+Nom : correct = Noun+A3sg+Pnon+Acc
2 : Noun+A3sg+P3sg+Nom : pre : Num+Card Noun+A3sg+P3sg+Nom : correct = Noun+A3sg+Pnon+Acc
2 : Noun+A3sg+P3sg+Nom : fors : Conj Verb+Pos+Past+A1sg : correct = Noun+A3sg+Pnon+Acc

2 : Noun+A3sg+P3sg+Nom : fors : Conj Noun+A3sg+Pnon+Acc : correct = Noun+A3sg+Pnon+Acc
2 : Noun+A3sg+P3sg+Nom : fors : Verb+Pos+Fut+A3sg Ques+Pres+A1pl : correct = Noun+A3sg+Pnon+Acc
2 : Noun+A3sg+P3sg+Nom : fors : Adverb Verb+Pos+Past+A3pl : correct = Noun+A3sg+Pnon+Acc
2 : Noun+A3sg+P3sg+Nom : fors : Num+Range Adj : correct = Noun+A3sg+Pnon+Acc
2 : Noun+A3sg+P3sg+Nom : fors : Verb+Pos+Fut+A1pl Punc : correct = Noun+A3sg+Pnon+Acc
2 : Noun+A3sg+P3sg+Nom : fors : Conj Adverb : correct = Noun+A3sg+Pnon+Acc
2 : Noun+A3sg+P3sg+Nom : fors : Pron+A1sg+Pnon+Acc Adj : correct = Noun+Prop+A3sg+Pnon+Nom
1 : Noun+A3sg+P3sg+Nom : fors : Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Abl Postp : correct = Noun+A3sg+Pnon+Acc
1 : Noun+A3sg+P3sg+Nom : fors : Verb^DB+Verb+Pass+Pos^DB+Noun+Inf2+A3sg+P3sg+Gen Noun+A3sg+P3sg+Abl : correct = Noun+A3sg+Pnon+Acc
1 : Noun+A3sg+P3sg+Nom : fors : Verb+Pos+Past+A3sg Noun+Prop+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Acc
1 : Noun+A3sg+P3sg+Nom : fors : Verb^DB+Verb+Caus+Pos^DB+Adverb+AfterDoingSo Noun+Prop+A3sg+Pnon+Acc : correct = Noun+A3sg+Pnon+Acc
1 : Noun+A3sg+P3sg+Nom : fors : Noun+A3sg+Pnon+Nom Verb+Pos^DB+Adj+Agt : correct = Noun+A3sg+Pnon+Acc
1 : Noun+A3sg+P3sg+Nom : fors : Postp Verb+Pos^DB+Adj+FutPart^DB+Noun+Zero+A3sg+P3sg+Nom : correct = Noun+A3sg+Pnon+Acc
1 : Noun+A3sg+P3sg+Nom : fors : Noun+A3sg+Pnon+Loc Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Dat : correct = Noun+A3sg+Pnon+Acc
1 : Noun+A3sg+P3sg+Nom : fors : Verb+Pos^DB+Noun+Inf2+A3pl+P3pl+Nom Postp : correct = Noun+A3sg+Pnon+Acc
1 : Noun+A3sg+P3sg+Nom : pre : Verb+Pos^DB+Noun+Inf2+A3sg+Pnon+Nom^DB+Adj+With Conj : correct = Noun+A3sg+Pnon+Acc
1 : Noun+A3sg+P3sg+Nom : pre : Noun+A3sg+P3sg+Loc Noun+Prop+A3sg+Pnon+Nom : correct = Noun+A3sg+Pnon+Acc
1 : Noun+A3sg+P3sg+Nom : pre : Conj Verb+Pos+Past+A3sg : correct = Noun+A3sg+Pnon+Acc

APPENDIX 4: LATTICE OF TURKISH

```
ANY
AdjMdfy ANY
Pron ANY
Noun ANY
Prop Noun
ADJ-SUF ANY
PRON-SUF ANY
Reflex PRON-SUF
ADVERB-SUF ANY
VERB-SUF ANY
VERB-TENSE VERB-SUF
NOUN-SUF ADJ-SUF ANY
NOUN-AGREEMENT NOUN-SUF
VERB-AGREEMENT VERB-SUF
A3pl NOUN-AGREEMENT VERB-AGREEMENT
A3sg NOUN-AGREEMENT VERB-AGREEMENT
NOUN-POSSESIVE NOUN-SUF
Pnon NOUN-POSSESIVE
P1sg NOUN-POSSESIVE
P2sg NOUN-POSSESIVE
P3sg NOUN-POSSESIVE
P1pl NOUN-POSSESIVE
P2pl NOUN-POSSESIVE
P3pl NOUN-POSSESIVE
NOUN-CASE NOUN-SUF
Acc NOUN-CASE
Dat NOUN-CASE
Loc NOUN-CASE
Abl NOUN-CASE
Gen NOUN-CASE
Ins NOUN-CASE
Nom NOUN-CASE
Adj ANY
Conj ANY
Verb ANY
Pres VERB-TENSE
Past VERB-TENSE
Aor VERB-TENSE
Fut VERB-TENSE
Narr VERB-TENSE
Prog1 VERB-TENSE
Prog2 VERB-TENSE
Neces VERB-TENSE
Opt VERB-TENSE
Imp VERB-TENSE
Desr VERB-TENSE
Cond VERB-TENSE
Cop VERB-TENSE
ASPECT*PR-CONT VERB-TENSE
A1sg VERB-AGREEMENT
A2sg VERB-AGREEMENT
A1pl VERB-AGREEMENT
A2pl VERB-AGREEMENT
VERB-SENSE VERB-SUF
Pos VERB-SENSE
Neg VERB-SENSE
Punc ANY
Interj ANY
NUMBER ANY
```

```
Num NUMBER
Card NUMBER
Ord NUMBER
Dist NUMBER
NOUN-DB ANY
POSTP ANY
Postp POSTP
PCNom POSTP
PCAbl POSTP
PCDat POSTP
PCIns POSTP
PCGen POSTP
Adverb ANY
PRON-DB ANY
PRON-DB-ADJ PRON-DB
Ques ANY
ADJ-DB ANY
ADJ-DB-NOUN ADJ-DB
NOUN-DB ANY
NOUN-DB-NOUN NOUN-DB
NOUN-DB-PRON NOUN-DB
NOUN-DB-ADJ NOUN-DB
NOUN-DB-ADVERB NOUN-DB
NOUN-DB-VERB NOUN-DB
PRON-DB-NOUN PRON-DB
PRON-DB-PRON PRON-DB
PRON-DB-VERB PRON-DB
ADVERB-DB ANY
ADVERB-DB-ADVERB ADVERB-DB
VERB-DB ANY
VERB-DB-ADVERB VERB-DB
VERB-DB-VERB VERB-DB
VERB-DB-NOUN VERB-DB
VERB-DB-ADJ VERB-DB
ADJ-DB-ADJ ADJ-DB
ADJ-DB-ADVERB ADJ-DB
ADJ-DB-VERB ADJ-DB
^DB+Noun+Zero ADJ-DB-NOUN
^DB+Adverb+Ly ADJ-DB-ADVERB
^DB+Verb+Zero ADJ-DB-VERB NOUN-DB-VERB PRON-DB-VERB
^DB+Adj+Rel PRON-DB-ADJ NOUN-DB-ADJ PRON-DB-ADJ
^DB+Adj+FitFor ADJ-DB-ADJ NOUN-DB-ADJ
^DB+Noun+Ness NOUN-DB-NOUN ADJ-DB-NOUN
^DB+Noun+Agt ADJ-DB-NOUN NOUN-DB-NOUN
^DB+Adj+Agt ADJ-DB-ADJ NOUN-DB-ADJ VERB-DB-ADJ
^DB+Noun+Dim ADJ-DB-NOUN
^DB+Verb+Become ADJ-DB-VERB NOUN-DB-VERB
^DB+Verb+Acquire ADJ-DB-VERB NOUN-DB-VERB
^DB+Adj+Without ADJ-DB-ADJ NOUN-DB-ADJ PRON-DB-ADJ
^DB+Adj+With ADJ-DB-ADJ NOUN-DB-ADJ PRON-DB-ADJ
^DB+Adverb+AsIf NOUN-DB-ADVERB VERB-DB-ADVERB
^DB+Adverb+Ly NOUN-DB-ADVERB
^DB+Noun+Dim NOUN-DB-NOUN
^DB+Pron+Rel PRON-DB-PRON NOUN-DB-PRON
^DB+Adverb+While ADVERB-DB-ADVERB VERB-DB-ADVERB
^DB+Verb+Caus VERB-DB-VERB
^DB+Verb+Pass VERB-DB-VERB
^DB+Verb+Recip VERB-DB-VERB
^DB+Verb+AbleNeg VERB-DB-VERB
^DB+Verb+Repeat VERB-DB-VERB
^DB+Verb+Hastily VERB-DB-VERB
```

```
^DB+Verb+EverSince VERB-DB-VERB
^DB+Verb+Able VERB-DB-VERB
^DB+Verb+Almost VERB-DB-VERB
^DB+Verb+Stay VERB-DB-VERB
^DB+Verb+Start VERB-DB-VERB
^DB+Noun+NotState VERB-DB-NOUN
^DB+Adverb+"maksizin" VERB-DB-ADVERB
^DB+Adverb+WithoutHavingDone VERB-DB-ADVERB
^DB+Noun+Inf1 VERB-DB-NOUN
^DB+Noun+Inf2 VERB-DB-NOUN
^DB+Noun+Inf3 VERB-DB-NOUN
^DB+Noun+FeelLike VERB-DB-NOUN
^DB+Adj+Zero VERB-DB-ADJ
^DB+Adj+PresPart VERB-DB-ADJ
^DB+Adj+FutPart VERB-DB-ADJ
^DB+Adj+FeelLike VERB-DB-ADJ
^DB+Adj+PastPart VERB-DB-ADJ
^DB+Adverb+AsLongAs VERB-DB-ADVERB
^DB+Adverb+When VERB-DB-ADVERB
^DB+Adverb+ByDoingSo VERB-DB-ADVERB
^DB+Adverb+AfterDoingSo VERB-DB-ADVERB
^DB+Adverb+SinceDoingSo VERB-DB-ADVERB
```

APPENDIX 5: LATTICE OF KAZAKH LANGUAGE

```
ANY
PRON-SUF ANY
NOUN ANY
POSTP ANY
NUMERAL ANY
ADJ-SUF ANY
ADVERB ANY
VERB-SUF ANY
PUNCTUATION ANY
Noun ANY
Verb ANY
Adverb ANY
Conj ANY
Adj ANY
Ques ANY
Prop ANY
PERSON-PRON PRON-SUF
Pron PRON-SUF
REFLEXIVE-PRON PRON-SUF
INDICATIVE-PRON PRON-SUF
INTERROGATIVE-PRON PRON-SUF
DEFINITE-PRON PRON-SUF
INDEFINITE-PRON PRON-SUF
NEGATIVE-PRON PRON-SUF
PersP PERSON-PRON
RefP REFLEXIVE-PRON
IndcP INDICATIVE-PRON
Intrg INTERROGATIVE-PRON
Indef DEFINITE-PRON
Negat INDEFINITE-PRON
Def NEGATIVE-PRON
NOUN-SUF ANY
NOUN-SUF ADJ-SUF ANY
NOUN-SUF PRON-SUF ANY
NOUN-AGR NOUN-SUF
NOUN-POSS NOUN-SUF
NOUN-CASE NOUN-SUF
A3sg NOUN-AGR VERB-AGR
A3pl NOUN-AGR VERB-AGR
Pnon NOUN-POSS
P1sg NOUN-POSS
P2sg NOUN-POSS
P2psg NOUN-POSS
P3sg NOUN-POSS
P1pl NOUN-POSS
P2pl NOUN-POSS
P2ppl NOUN-POSS
P3pl NOUN-POSS
Nom NOUN-CASE
Ins NOUN-CASE
Gen NOUN-CASE
Dat NOUN-CASE
Abl NOUN-CASE
Loc NOUN-CASE
Acc NOUN-CASE
NUM-SUF ANY
Num NUM-SUF
Card NUM-SUF
Ord NUM-SUF
```

```
Coll NUM-SUF
Aprx NUM-SUF
ADJ-SUF ANY
COMP-ADJ ADJ-SUF
CompP COMP-ADJ
CompN COMP-ADJ
VERB-SUF ANY
VERB-SENSE VERB-SUF
Neg VERB-SENSE
Pos VERB-SENSE
VERB-TENSE VERB-SUF
PastNRINDF VERB-TENSE
PastDF VERB-TENSE
CondMD VERB-TENSE
ImpMD VERB-TENSE
OptMD VERB-TENSE
PresGO VERB-TENSE
FutTR VERB-TENSE
PresDF VERB-TENSE
FutINDF VERB-TENSE
FutGO VERB-TENSE
PastTR VERB-TENSE
PastNRDF VERB-TENSE
PresPR VERB-TENSE
Pres VERB-TENSE
VERB-AGR VERB-SUF
WISH-CLAUSE VERB-SUF
A1sg VERB-AGR
A2sg VERB-AGR
A2psg VERB-AGR
A1pl VERB-AGR
A2pl VERB-AGR
A2ppl VERB-AGR
WishClause WISH-CLAUSE
Postp POSTP
PCNom POSTP
PCAbl POSTP
PCDat POSTP
PCIns POSTP
PCGen POSTP
PRON-DB ANY
ADJ-DB ANY
NOUN-DB ANY
ADVERB-DB ANY
VERB-DB ANY
NUM-DB ANY
NUM-DB-NOUN NUM-DB
PRON-DB-ADJ PRON-DB
NOUN-DB-NOUN NOUN-DB
NOUN-DB-PRON NOUN-DB
NOUN-DB-ADJ NOUN-DB
NOUN-DB-ADVERB NOUN-DB
NOUN-DB-VERB NOUN-DB
PRON-DB-NOUN PRON-DB
PRON-DB-PRON PRON-DB
PRON-DB-VERB PRON-DB
ADVERB-DB-ADJ ADVERB-DB
VERB-DB-ADVERB VERB-DB
VERB-DB-VERB VERB-DB
VERB-DB-NOUN VERB-DB
VERB-DB-ADJ VERB-DB
```

```
ADJ-DB-ADJ ADJ-DB
ADJ-DB-ADVERB ADJ-DB
ADJ-DB-VERB ADJ-DB
ADJ-DB-NOUN ADJ-DB
^DB+Noun+Zero ADJ-DB-NOUN NUM-DB-NOUN PRON-DB-NOUN
^DB+Adj+Zero ADJ-DB-ADJ NOUN-DB-ADJ
^DB+Pron+Poss NOUN-DB-PRON
^DB+Verb+Zero NOUN-DB-VERB
^DB+Noun+Agt1 NOUN-DB-NOUN
^DB+Noun+Agt2 NOUN-DB-NOUN
^DB+Noun+Small1 NOUN-DB-NOUN
^DB+Noun+Small2 NOUN-DB-NOUN
^DB+Noun+Theme NOUN-DB-NOUN
^DB+Noun+Group NOUN-DB-NOUN
^DB+Noun+Rel1 NOUN-DB-NOUN
^DB+Noun+Rel2 NOUN-DB-NOUN
^DB+Adj+Rel NOUN-DB-ADJ
^DB+Noun+Place NOUN-DB-NOUN
^DB+Adj+FitFor NOUN-DB-ADJ
^DB+Adj+Group NOUN-DB-ADJ
^DB+Adj+Between NOUN-DB-ADJ
^DB+Adj+Agt2 NOUN-DB-ADJ
^DB+Adj+Agt1 NOUN-DB-ADJ
^DB+Adj+With NOUN-DB-ADJ
^DB+Adj+qor NOUN-DB-ADJ
^DB+Adj+qoy NOUN-DB-ADJ
^DB+Adj+Issue NOUN-DB-ADJ
^DB+Adj+SeemAS NOUN-DB-ADJ
^DB+Adj+As NOUN-DB-ADJ
^DB+Adj+As1 NOUN-DB-ADJ
^DB+Verb+Do NOUN-DB-VERB
^DB+Verb+Perform NOUN-DB-VERB
^DB+Verb+SeemAs NOUN-DB-VERB
^DB+Adverb+AccordingTo1 NOUN-DB-ADVERB
^DB+Adverb+AccordingTo2 NOUN-DB-ADVERB
^DB+Adverb+As1 NOUN-DB-ADVERB
^DB+Adverb+As2 NOUN-DB-ADVERB
^DB+Adverb+Zero NOUN-DB-ADVERB
^DB+Adverb+InThatWay ADJ-DB-ADVERB
^DB+Adverb+InThatTime ADJ-DB-ADVERB
^DB+Adverb+InThatCause ADJ-DB-ADVERB
^DB+Verb+sIn ADJ-DB-VERB
^DB+Verb+Ay ADJ-DB-VERB
^DB+Verb+Ip ADJ-DB-VERB
^DB+Verb+PA ADJ-DB-VERB
^DB+Adj+Time ADVERB-DB-ADJ
^DB+Verb+Act ADVERB-DB-VERB NOUN-DB-ADJ
^DB+Verb+Ref VERB-DB-VERB
^DB+Verb+CausV VERB-DB-VERB
^DB+Verb+Pass1 VERB-DB-VERB
^DB+Verb+Pass2 VERB-DB-VERB
^DB+Verb+Coll VERB-DB-VERB
^DB+Verb+CausT VERB-DB-VERB
^DB+Verb+CausD VERB-DB-VERB
^DB+Adverb+AsFarAs VERB-DB-ADVERB
^DB+Adverb+ByDoingSo VERB-DB-ADVERB
^DB+Adverb+ByDoningSo VERB-DB-ADVERB
^DB+Adverb+AfterDoningSo VERB-DB-ADVERB
^DB+Adverb+HabPart VERB-DB-ADVERB
^DB+Adverb+As VERB-DB-ADVERB
^DB+Adverb+InsteadOf VERB-DB-ADVERB
```

```
^DB+Adj+HabPart VERB-DB-ADJ
^DB+Adj+FutPart VERB-DB-ADJ
^DB+Adj+PastPart VERB-DB-ADJ
^DB+Adj+Without VERB-DB-ADJ NOUN-DB-ADJ
^DB+Noun+Inf VERB-DB-NOUN
^DB+Noun+FutPart VERB-DB-NOUN
^DB+Noun+PastPart VERB-DB-NOUN
^DB+Noun+Position VERB-DB-NOUN
^DB+Noun+WellDone VERB-DB-NOUN
^DB+Noun+Item VERB-DB-NOUN
^DB+Noun+Person VERB-DB-NOUN
^DB+Noun+Job VERB-DB-NOUN
^DB+Noun+Place VERB-DB-NOUN
^DB+Noun+Item VERB-DB-NOUN
```

APPENDIX 6: TENSES AND THEIR PROPERTIES IN KAZAKH LANGUAGE

1-Болжамды Келер Шақ (Future Indefinite Tense)

мен (I)	P1Sg			MJн
біз (we)	P1Pl			MJз
сен (you)	P2Sg			cJң
сендер (you *)	P2Pl		**+ар**	cJңдАр
сіз (you+)	P2PSg	**Verb**	**+ер**	cJз
сіздер (you+*)	P2PPl		**+р**	cJздАр
ол (he,she)	P3Sg			**-**
олар (they)	P3Pl			
(М→м,п,б), (J→ы, і), (А→а,е)				

1- Conserned of actions which are assumed to be performed in future.

2- If verb ends with consonant letter and vowel of last syllable is one of BackVowel then suffix is **–ар** (ar), if vowel of last syllable is one of FrontVowel then suffix is **–ер** (er) and if verb ends with vowel letter then suffix is **–р** (r) . For example:
Мен ертең дүкенге бар-ар-мын. (men erteN dUkenge bararmIn) -*I will probably go to market tomorrow*

3- Negative form is made by adding negation suffixes **ба, бе** (ba/be), **па, пе** (pa/pe), **ма, ме** (ma/me) and after by adding suffix as **–с** (s) with proper personal agreement.
For example:
Мен ертең дүкенге бар-ма-с-пын. (men erteN dUkenge barmaspIn)-*I will not probably go to market tomorrow*

4- RULE B, RULE M, RULE J and RULE A are applicable.

2- Мақсатты Келер Шақ (Future Goal Oriented Tense)

мен (I)	P1Sg				MJн
біз (we)	P1Pl			(Optional)	MJз
сен (you)	P2Sg				cJң
сендер (you *)	P2Pl				cJңдАр
сіз (you+)	P2PSg	**Verb**	**+MAQ**	**+шJ**	cJз
сіздер (you+*)	P2PPl				cJздАр

ол (he,she) олар (they)	P3Sg P3Pl				-
(M→м,п,б), (J→ы, і), (A→а,е), (Q→қ, к)					

1- Conserdned about actions which is planned to be performed in future. For example:
Мен жаз-бақ-пын. -(men Zaz-baq-pIn)- *I am planning to write*

2- If action is in more decided form than suffixes as SI/Si are added.

For example: **Мен жаз-бақ-шы-мын.** (men Zazbaq-SI-mIn)-*I am going to write or I intend to write*

3- Negative form is constructed by negative auxilary verb **ЕМЕС** with proper personal agreement. For example: **Мен жаз-бақ-шы емеспін.** (men Zazbaq-SI emespin)-*I am not going to write*

4- Without negative auxilary verb it can be occur in part participle form which define noun.

5- RULE Q, RULE M, RULE J and RULE A are applicable.

3-Ауыспалы Келер Шақ (Future Transitional)

мен (I) біз (we)	P1Sg P1Pl			мJн мJз
сен (you) сендер (you *) сіз (you+) сіздер (you+*)	P2Sg P2Pl P2PSg P2PPl	**Verb**	**+а** **+е** **+й**	сJң сJңдАр сJз сJздАр
ол (he,she) олар (they)	P3Sg P3Pl			дJ
(J→ы, і), (A→а,е)				

1- This tense is between present and future tense for that reason it is transitional. Future or present meaning of sentence is identified semantically.
2- Verbs are used without infinity suffix such as **-у** after verbal roots and letters -а**,-е,-й** are added with proper agreement.
   a) **-а** added to word which contains Back Vowels and ends with Consonant
   b) **-е** added to word which contains Front Vowels and ends with Consonant
   c) **-й** added to word which ends with Vowel.

For example: **Мен жаз-а-мын.** (men Zaz-a-mIn) –*I will write*

Сен қазақша сөйле-й-сің. (sen qazaqSa sOyle-y-siN) **–** *You will speak in Kazakh*

3- Verbs with infinity forms of ending –ю (Y) such as
   a) **қою** (qoY)-*to put*
   b) **сою** (soY)-*to kill animal*
   c) **құю** (quY)-*to pour*
   d) **жаю** (ZaY) -*to strech*
   e) **шаю** (SaY)- *to rinse*
   f) **сүю** (sUY)-*to love*

In fact verb **қою** (qoY) has infinity form **қойу** (qoy-w) and after infinity suffix **–у** (w) is dropped remained verb is as **қой** (qoy). Therefore tenses suffix which is **–а (a)** is added to verb as **қой-а-мын** (qoy-a-mIn), rule of substituting of **й+а=я –йа** (ya) with **-я** (A) named "RULE Y" will occur. Then it becomes **қоя-мын** (qoA-mIn). However with verb **сүю** (sUY)-*to love*, there is any conversion happens thus it remains as it was. For Example:

Мен сені сүй-е-мін. (men seni sUy-e-min) – *I love you*

Мен кофе құямын. (men kofe quA-mIn) **–** *I pour a coffee*

4- With verbs like
   a) **оқу** (oqw)**-** *to read*
   b) **есту** (estw)**-** *to hear*
   c) **ренжу** (renZw)**-** *to be offended*
   d) **даму** (damw)- *to improve*
   e) **қобалжу** (qobalZw)- *to worry*
   f) **тану** (tanw)- *to recognize*
   g) **еру** (erw)- *to follow*
   h) **абыржу** (abIrZw)- *to be concerned*

   Here, when infinity affix **–у** is dropped, vowels such as **ы** or **і** are remained and when tense suffix is added which is **–й**, substitution rule of **(ы+і= и)** named "RULE W" occurs. For example:

   a) **Мен оқ-и-мын.** (oqw → men oq-j-mIn) – *I will read*
   b) **Мен ест-и-мін.** (estw → men est-j-min) – *I will hear*

5- Rule of CHTS(Converting Hard Letters to Soft) is valid here.
6- This tense is used in fairy tails to tell about some actions happened in very past.
   For Example: **Кішкентай қонақ үйге бір жалмауыз кемпір кел-е-ді.** (kiSkentay qonaq Uyge bir Zalmawz kempir keledi) – *A witch came to little hall*

7- Negative form is produced by negation affixes **ба, бе** (ba/be), **па, пе** (pa/pe), **ма, ме** (ma/me) and after letter **–й** (y) is added to verbs.Verbs with infinity forms of ending –ю (Y) such as

a) **қою** (qoY)-*to put*
b) **сою** (soY)-*to kill animal*
c) **құю** (quY)-*to pour*
d) **жаю** (ZaY) *–to strech*
e) **шаю** (SaY)- *to rinse*
f) **сүю** (sUY)-*to love* after dropped infinity affix **-у** (w) the letter **–й** (y) is remained and negation affixes are added.
For example: **мен қой-ма-ймын.** (men qoy-ma-ymIn) -*I will not put*

With verbs like:

g) **оқу oqw-** *to read*
h) **есту estw-** *tohear*
i) **ренжу renZw-** *to be offended*
j) **даму damw-** *to improve*
k) **қобалжу qobalZw-** *to worry*
l) **тану tanw-** *to recognize*
m) **еру erw-** *to follow*
n) **абыржу ablrZw-** *to be concerned*

after dropped infinity suffix **– у** (w) the letters **ы or і** (I or i) are remained and negation suffixes are added whereafter. For example: **мен ест-і-ме-ймін.** (men estimeymin)- *I will not hear*

Interrogative forms are made by affixes **ба, бе** (ba/be), **па, пе** (pa/pe), **ма, ме** (ma/me) after verbs with proper personal agreements. Here A3Sg and A3Pl sometimes have optionality of adding dI/di suffixes. For example: **Алма театрға бара(ды) ма?** (alma teatrGa bara(dI) ma?)-*Will Alma go to a theater?*

4-**Жедел Өткен Шақ (Past Definite Tense)**

мен (I)	P1Sg	**Verb**	**+DJ+**	м
біз (we)	P1Pl			Q
сен (you)	P2Sg			ң
сендер (you *)	P2Pl			ңдАр
сіз (you^{+})	P2PSg			ңJз
сіздер (you^{+*})	P2PPl			ңJздАр
ол (he,she)	P3Sg			-
олар (they)	P3Pl			

(J→ы, i), (A→a,e), (Q→қ, к), (D→д,т)

1- Conserned of actions which are definitely performed in past.
2- Infinity suffix of verb as **-у** (w) is dropped and dJ suffix of Past Definite tense is added with proper personal agreement.
   a) RULE D is valid for adding suffixes of tense to verbs
      мен кет-ті-м (men ket-ti-m)- *I left*

      мен жаз-ды-м (men Zaz-dI-m)- *I left*

   b) RULE J, RULE A are valid for Vowel Harmony

3- RULE Y! is valid here with verbs having last letter of Y in infinite form.
   For example:

   мен құй-ды-м. (men quy-dIm)- *I poured*

   мен жой-ды-м. (men Zoy-dIm)- *I cancelled*

4- Negative form is made by adding negation suffixes **ба, бе** (ba/be), **па, пе** (pa/pe), **ма, ме** (ma/me) and after by adding tense suffix which is **DJ** with proper personal agreement.
5- RULE D, RULE W!, RULE Y! And RULE Q are valid.

5-Айғақты Бұрыңғы Өткен Шақ (Past Narrative Definite Tense)

мен (I)	P1Sg	**Verb**		мJн
біз (we)	P1Pl			бJз
сен (you)	P2Sg		**+GAн+**	сJң
сендер (you *)	P2Pl			сJңдAр
сіз (you^{+})	P2PSg		**+QAн+**	сJз
сіздер (you^{+*})	P2PPl			сJздAр
ол (he,she)	P3Sg			-
олар (they)	P3Pl			
(J→ы, i), (A→a,e), (Q→қ, к), (G→ғ,г)				

1- Spesifies the time of action which had been done in past with witnesses.
2- Suffixes **-ған -ген** (Gan/gen) , **-қан, -кен** (qan/ken) are added to verbs which are used without infinity suffix such as **–у** after verbal roots then added proper personal agreements.

For example:

a) **Біз ауылға хат жібер-ген-біз** (biz awlIGa xat Zibergenbiz)-*It is known that we sent a letter to village*
b) **сен кеше дүкенге бар-ған-сың** (sen keSe dUkenge bar-Gan-sIN)-*It is known that you went to market yesterday*

3- RULE Y! is applicable for related verbs.
For example:

қою → қойу →қой-ған (qoY → qoyw → qoy-Gan)-*to put*

4- RULE W! is applicable for related verbs.
For example:

оқу→ оқы → оқы-ған (oqw → oqI → oqI-Gan)-*to read*

5- Negative form constructed in two ways:

a. negation suffixes **ба, бе** (ba/be), **па, пе** (pa/pe), **ма, ме** (ma/me) added to verbs and after that tense suffix is added
b. after verb is taken tenses suffix and then with negative auxilary verbs such as **жоқ** (Zoq) **, емес** (emes)

6- RULE Q, RULE G, RULE W!, RULE Y!, RULE A And RULE J are applicable.

6-Айғақсыз Бұрыңғы Өткен Шақ (Past Narrative Indefinite Tense)

<table>
<tr><td>мен (I)
біз (we)</td><td>P1Sg
P1Pl</td><td rowspan="5">Verb</td><td rowspan="5">+ып+
+іп+
+п+</td><td>пJн
пJз</td></tr>
<tr><td>сен (you)
сендер (you *)
сіз (you+)
сіздер (you+*)</td><td>P2Sg
P2Pl
P2PSg
P2PPl</td><td>сJң
сJңдАр
сJз
сJздАр</td></tr>
<tr><td>ол (he,she)
олар (they)</td><td>P3Sg
P3Pl</td><td>тJ</td></tr>
<tr><td colspan="4">(J→ы, і), (А→а,е)</td></tr>
</table>

1- Defines the time of action which had been done in past without witnesses or indefinite time of action.

2- Suffixes **–ып, -іп, -п (** Ip / ip / p) are added to verbs with proper personal agreement. For example: **сіз ауылға хат жібер-іп-сіз.** (siz awlIGa xat Ziberipsiz)-*You probably send a letter to village*

3- Negative form is constructed by negation suffixes ba/be, pa/pe, ma/me added to verbs and after that tense suffix -p(п) added to verbs.

4- RULE V is valid. For example:
тебу → теуіп (tebw→ tew-ip) *-to ride*

5- RULE J, RULE H, RULE A and RULE V are applicable.

For example: **Ол өзінің досымен кездес-пе-п-ті** (ol OziniN dosImen kezdes-pe-p-ti) *–He probably did not meet with his friend.*

7-Ауыспалы Өткен Шақ (Past Transitional Tense)

мен (I)	P1Sg	**Verb**		мJн
біз (we)	P1Pl			бJз
сен (you)	P2Sg		**+АтJн+**	сJң
сендер (you *)	P2Pl			сJңдАр
сіз (you+)	P2PSg		**+йтJн+**	сJз
сіздер (you+*)	P2PPl			сJздАр
ол (he,she)	P3Sg			-
олар (they)	P3Pl			
(J→ы, i), (A→a,e)				

1- Defines the time of action which is continuously or regularly has been done in past
2- Suffixes **–АтJн, -йтJн** are added to verbs with proper personal agreements.
3- RULE J and RULE A are valid.
4- For example:
біз мына үйде тұратынбыз. (biz mIna Uyde turatInbIz)- *We had been living in this house*
5- RULE Y is valid for this tense.
For example:

қою→ қойу→қой-а-тын → қоя-тын (qoY→ qoyw→ qoy-atIn → qoA-tIn) **-***to put*

6- RULE W is valid for this tense.

For example:

оқу→ оқы → оқы-йтын → оқ-и-тын (oqw → oql → oql-ytln → oqj-tln) **-***to read*

7- Negative form is constructed by adding negation suffixes such as **ба, бе** (ba/be), **па, пе** (pa/pe), **ма, ме** (ma/me) added to verbs and after that tense suffix AtJn is added with proper personal agreements.

8-Мақсатты Осы Шақ (Present Goal Oriented Tense)

мен (I)	P1Sg	**Verb**	**+QAлJ** **+GAлJ**	Auxilary Verbs **тұр** **жүр** **отыр** **жатыр**	мJн
біз (we)	P1Pl				мJз
сен (you)	P2Sg				сJң
сендер (you *)	P2Pl				сJңдAр
сіз (you+)	P2PSg				сJз
сіздер (you+*)	P2PPl				сJздAр
ол (he,she)	P3Sg				-
олар (they)	P3Pl				
(М→м,п,б), (J→ы, і), (А→а,е), (Q→қ, к), (G→ғ, г)					

1- Conserned about actions which is planned to be performed at that moment.
For example:
Мен жаз-ғалы тұрмын.(**men Zaz-Gall turmln)**-*I am planning to write now*
2- Suffixes **– GaлJ, -QAлJ** added to verb and with suitable auxilary verb formed with proper tense and personal agreement.

3- If auxilary is not used then this can be occured as deverbal adverb which refers to an adverb derived from a verb.
For example:

Сіз айтқалы мен ойландым.(siz ayt-qall men oylandlm) -*I thought after that you said*

4- Negative form is constructed by negation suffixes **ба, бе** (ba/be), **па, пе** (pa/pe), **ма, ме** (ma/me) and after suffixes are added to verbs with suitable auxilary verb with proper personal agreement.

9-Дәл Осы Шақ (Present Progressive Tense)

мен (I)	P1Sg			мJн
біз (we)	P1Pl			мJз
сен (you)	P2Sg			сJң
сендер (you *)	P2Pl			сJңдAр
сіз (you+)	P2PSg	**Verb**	**+DA+**	сJз
сіздер (you+*)	P2PPl			сJздAр
ол (he,she)	P3Sg			-
олар (they)	P3Pl			
(J→ы, і), (A→a,e), (D→д,т)				

1- Conserned about actions which is on progress at that moment.
2- Suffixes -DA added to verbs in infinity form and with suitable personal agreement.

For example:

мен жазу-да-мын. (men Zazw-da-mln)-*I am writing now*

3- RULE A is valid.
For example:

сіз айту-да-сыз. (siz aytw-da- slz) -*You are saying*

мен тазалау- да-мын. (men tazalaw-da-mln)-*I am cleaning*

сен кездесу-де-сің. (sen kezdesw- de-siN) -*you are meeting*

4- Negative form is constructed by negation suffixes **ба, бе** (ba/be), **па, пе** (pa/pe), **ма, ме** (ma/me) added to infinity form of verb and ended with proper personal agreement.
For example:

сіз айт-пау-да-сыз. (siz ayt-pa-wda- slz) -*You are not saying*

мен тазала-мау-да-мын. (men tazala-maw-da-mln)-*I am not cleaning*

сен кездес-пеу-де-сің. (sen kezdes-pew-de-siN) -*You are not meeting*

5- In Negative case RULE Y! is valid.
For example:

мен құй-мау-да-мын. (men quy-maw-damln)-*I am not pouring*

10-Нақ Осы Шақ (Present Definite Tense)

<table>
<tr><td>мен (I) P1Sg
біз (we) P1Pl</td><td rowspan="3">Verb</td><td rowspan="3">+ып
+іп
+п</td><td rowspan="3">Auxilary Verbs
тұр
жүр
отыр
жатыр</td><td>мJн
мJз</td></tr>
<tr><td>сен (you) P2Sg
сендер (you *) P2Pl
сіз (you+) P2PSg
сіздер (you+*) P2PPl</td><td>сJң
сJңдАр
сJз
сJздАр</td></tr>
<tr><td>ол (he,she) P3Sg
олар (they) P3Pl</td><td>-</td></tr>
<tr><td colspan="5">(J→ы, і), (А→а,е)</td></tr>
</table>

1. Conserned about actions which is on progress at that moment according to physical condition of doer.

Auxilary Verbs	**Position of Object while Performing Main Verb**	**Period of Performed Main Verb**	**Usage**
тұр- stand	in stand position	very short	At the moment
жүр - go	in action position	short	Continuously
отыр -sit	in sitting position	long	Only with living things
жатыр -lie	in lieing position	very long	Throughout activity

2. Suffixes **–ып, -іп, -п** added to verb in **INF0FORM** and with suitable auxilary verb formed with proper tense and personal agreement.

1- verbs which end with **б**, here **б** is substituted by **у**.
For example:

a) **жабу** → **жауып** (Zabw → Zawlp)- *to clos*e
b) **тебу** → **теуіп** (tebw→ tewip)- *to drive*
c) **табу** → **тауып** (tabw→ tawlp)- *to find*

2- verbs which end with **ю**, here it is substituted by **й**
For example:

a) **қою – қойып** (qoY→ qoylp *)-to put*
b) **сою – сойып** (soY→ soylp *)-to cut*

3- With verbs such as **бару, келу, апару, әкелу** only auxilary verb **жатыр** is used.
4- Negative form is formed in two ways:
    1) **ба, бе** (ba/be), **па, пе** (pa/pe), **ма, ме** (ma/me) negation suffixes are added to verbs and then the letter **й** is added. After suitable auxilary verb with proper agreement according to semantic meaning is added.For example :
    Мен қой-ма-й тұр-мын. (men qoy-ma-y tur-mln)-*I am not putting*

    2) **–ған, -ген, -қан, -кен** (qan/ken, Gan/gen) affixes are added to auxilary verbs and an other auxilary verb that is **жоқ** (Zoq) comes after with proper agreement.
    Мен қойып тұрған жоқпын. (men qoy-lp turGan Zoq-pln)-*I am not putting*

Transcription and Rules are given in Appendix 7 and Appendix 8.

APPENDIX 7: LATIN REPRESENTATION AND TRANSCRIPTION OF KAZAKH LETTERS

Kazakh Letters in Cyrillic	**а**	**ә**	**б**	**в**	**г**	**ғ**	**д**	**е**	**ж**	**з**	**и**	**й**	**к**	**қ**	**л**	**м**	**н**	**ң**	**о**
Transcription	ʌ	ə	b	v	g	ğ	d	e	ʒ	z	i:	i	k	q	l	m	n	ŋ	o
Latin Representation	a	**E**	b	v	g	G	d	e	Z	z	j	y	k	q	l	m	n	N	o

Kazakh Letters in Cyrillic	**ө**	**п**	**р**	**с**	**т**	**у**	**ұ**	**ү**	**ф**	**һ**	**х**	**ц**	**ч**	**ш**	**щ**	**ы**	**і**	**ю**	**я**
Transcription	ɜ:	p	r	**s**	t	ʊ	u:	u	f	h	eks	ts	tʃ	ʃ	ʃ ʃ	I	i	ju	ja
Latin Representation	O	p	r	**s**	t	w	u	U	f	h	x	**c**	**C**	S	H	I	i	Y	A

APPENDIX 8: TABLE OF ALTERNATION RULES OF KAZAKH LANGUAGE

Name of Rule	Explanation	Verbs Used	Examples
RULE INF0	Infinity Verb- y	all verbs	**жүру→жүр** ZUrw→ZUr *to walk→walk*
RULE Y	й + а = я	verbs which ends with й	**қой-а-мын→қоя-мын** qoy-a-mIn → qoA-mIn *I'll put*
RULE Y!	ю - у = й	verbs which ends with ю	**құю→құйу→құй-ды-м** quY→quyw→quydIm *I poured*
RULE A	[VLS] =BackVowel then A→a [VLS] =FrontVowel then A→e	verbs whose vowel of last syllable is backvowel or frontvowel	**жинау-да,** **істеу-де** Zjnaw-da, istew-de *collecting, doing*
RULE J	[VLS] =BackVowel then J→ы [VLS] =FrontVowel then J→i	verbs whose vowel of last syllable is backvowel or frontvowel	**есті-дJ → есті-ді** **жина-дJ→ жина-ды** esti-dJ→esti-di Zjna-dJ→ZjnadI *heard* *collected*
RULE W	ы + й = и і + й =и	some special verbs which end with ы or і	**оқы-йын → оқ-и-ын** oqI-yIn → oq-j-In *I shall read*
RULE W!	Verb – у = ы or і	reverse of RULE W	**оқу→оқ-ы-ды** oqw→oqI-dI *read*
RULE D	LL# = UnVCons then D→ т LL# = ![UnVCons] then	verbs whose last letter is unvoiced consonant	**кет-ті-м** **жаз-ды-м** ket-ti-m Zaz-dI-m

	D→ д		*I left, I wrote*
RULE V	Verb + Vowel+п then б → у	verbs which ends with б in **INF0** form	**жабу → жауып** Zabw→Zawlp *to close*
CHTS RULE	Verb + Vowel then к→ г қ→ ғ п→ б	verbs which ends with к, п, қ	**төк→ төгіп** **тық→ тығып** **жауап→ жауабы** tOk→ tOg-ip tIq→ tIG-Ip Zawap→ Zawab-I *to spill* *to hide* *answer*
RULE G	[VLS] = BackVowel LL# = VCons& SonCons then G→ ғ [VLS] = FrontVowel LL# = VCons& SonCons then G→ г	verbs whose vowel of last syllable is backvowel or frontvowel and last letter is voiced consonant or sonorous consonant	**жина→жина-ғ-ым** **кел→кел-г-ім** **Zjna→Zjna-G-Im** **kel→ kel-g-im** *to pick off* *to come*
RULE B	LL# = Cons then B→ A LL# = Vowel then B→ 0(nullor)	verbs whose last letter is consonant or vowel	**бару→бар-а-р** **келу→кел-е-р** **егеу→еге-р** barw→bar-a-r kelw→ kel-e-r egew→ ege-r *to arrive* *to come* *to sharpen*
RULE H	LL# = Cons then H→ J	verbs whose last letter is	**бару→бар-ы-п** **келу→кел-і-п**

	LL# = Vowel then H→ 0(nullor)	consonant or vowel	**еге→еге-п** barw→bar-l-p kelw→ kel-i-p egew→ ege-p *to arrive* *to come* *to sharpen*
RULE K	LL# = UnvCons then K→ к LL# = ![UnvCons] then K→ г	verbs whose last letter is unvoiced consonant or not unvoiced consonant	**заң-гер** **іс-кер** zaN-ger is-ker *lawyer* *laborer*
RULE Q	[VLS] = BackVowel then Q→ қ [VLS] = FrontVowel then Q→ к	verbs whose vowel of last syllable is backvowel or frontvowel	**барма-қ** **келме-к** bar-ma-q kel-me-k *going to arrive* *going to come*
RULE M	LL# = UnvCons then M→ п LL# = VCons& SonCons2 then M→ б LL# = Vowel& SonCons1 then M→ м	verbs whose last letter is unvoiced consonant, voiced consonant, Sonorous consonant or vowel	**бар-ма** **жаз-ба** **тап-па** bar-ma Zaz-ba tap-pa *shall not go* *shall not write* *shall not find*

- VLS - Vowel of last Syllable
- LL# - Last Letter of a Word

CURRICULUM VITAE

Credentials

Name, Surname	:	Gulshat KESSIKBAYEVA
Place of Birth	:	Taraz/KAZAKHSTAN
Marital Status	:	Married
E-mails	:	gulshatk@hacettepe.edu.tr shatik2030@gmail.com
Address	:	Department of Computer Engineering, No: 212, Hacettepe University, Beytepe, Ankara/TURKEY

Education

High School	:	Math and Physics-Almaty/KAZAKHSTAN
BSc.	:	Department of Mathematics-Education, Middle East Technical University, Ankara, TURKEY
MSc.	:	Department of Computer Engineering, Istanbul Technical University, Istanbul/ TURKEY
PhD.	:	Department of Computer Engineering, Hacettepe University, Ankara/TURKEY

Foreign Language

Kazakh (Native), Russian (Excellent), Turkish (Excellent), English (Advanced), Japanese (Intermediate)

Work Experience

Senior Lecturer in Department of Computer Engineering, Suleymen Demirel University, Kazakhstan.

Area of Experience

Natural Language Processing, Machine learning, Database and Software Cost Estimation, Operating and Distributed Systems

Project and Budgets

Publications

G. Kessikbayeva and I. Cicekli, “Rule Based Morphological Analyzer of Kazakh Language,” in *Proceedings of the 2014 Joint Meeting of SIGMORPHON and SIGFSM*, pp. 46–54, **2014**.

G. Kessikbayeva and I. Cicekli, “A Rule Based Morphological Analyzer and A Morphological Disambiguator for Kazakh Language”, Linguistics and Literature Studies, Vol. 4(1), pp. 96—104, **2016**

Oral and Poster Presentations