Computer Analysis of the Turkmen Language Morphology

A. Cüneyd Tantuğ[1], Eşref Adalı[1], and Kemal Oflazer[2]

[1] İstanbul Teknik Üniversitesi Elektrik-Elektronik Fakültesi
Bilgisayar Mühendisliği Bölümü
34469, Maslak, İstanbul, Türkiye
{cuneyd, adali}@cs.itu.edu.tr
[2] Sabancı Üniversitesi Doğa Bilimleri Fakültesi
Bilgisayar Mühendisliği Bölümü
34956, Orhanlı, Tuzla, Türkiye
oflazer@sabanciuniv.edu

Abstract. This paper describes the implementation of a two-level morphological analyzer for the Turkmen Language. Like all Turkic languages, the Turkmen Language is an agglutinative language that has productive inflectional and derivational suffixes. In this work, we implemented a finite-state two-level morphological analyzer for Turkmen Language by using Xerox Finite State Tools.

1 Introduction

This paper describes the implementation of a two-level morphological analyzer for the Turkmen Language. The Turkmen Language is classified as one of the Turkic languages which are all in the Ural-Altaic language family. Like all Turkic languages, the Turkmen Language is an agglutinative language that has productive inflectional and derivational suffixes which are affixed to a word like "beads on a string" [1]. Hence, the morphological analyzing component is an important first-step component of any natural language processing task for agglutinative languages which have complicated morphological structures. There are lots of work in the literature which have focused on two-level analysis of various languages like Japanese, English and Finnish [2,3,4] but the most valuable work for our developments is the two-level description of Turkish morphology [5]. Since both Turkish and Turkmen are close Turkic languages, some of the morphological structures and word roots are common. Even though both languages are similar, there exist great divergences like different tenses, different subject-verb agreements and etc. Another morphological work for a Turkic language is for Crimean Tatar [6].

In this work, we implemented a finite-state two-level morphological analyzer for Turkmen Language by using Xerox Finite State Tools [7]. Next section gives some information about the Turkmen Language; the following chapter describes a brief overview of two-level morphology. In the fourth section, the two-level rules are explained in detail while the finite state machines for morphotactics are explained in the fifth section. The last section concludes the results of the work.

T. Salakoski et al. (Eds.): FinTAL 2006, LNAI 4139, pp. 186 – 193, 2006.

2 The Turkmen Language

The Turkmen Language is a Turkic language which belongs to Ural-Altaic language family. It is used by nearly 7 million people especially in Turkmenistan, Afghanistan, Iran, and Iraq [8]. Although there are great similarities between Turkish and Turkmen language, these languages are classified as different languages because Turkmen is not intelligible to Turkish speaking people or vice versa. After using Arabic and Cyrillic alphabets, the current Turkmen orthography is composed of 30 Latin letters: ***a b j ç d e ä f g h y i ź k l m n ñ o ö p r s ş t u ü w ý z***. There are 9 vowels (***a e ä y i o ö u ü***) and 21 consonants (***b j ç d f g h ź k l m n ñ p r s ş t w ý z***).

3 Two-Level Morphology

Two-level morphology is a widely used technique in morphological analysis [9]. As the name emphasizes, there are two levels called lexical and surface levels. In the surface level, a word is represented in its original orthographic form. In the lexical level, a word is represented by denoting all of the functional components of the word. The phonological modifications can be implemented by writing rules these four rule types [1]:

1.	`a:b => LC _ RC`	A is realized as b only in the context LC (left context) and RC (Right Context), but not necessarily.
2.	`a:b <= LC _ RC`	A is **always** realized as b in the context LC and RC.
3.	`a:b <=> LC _ RC`	A is **always** realized as b in the context LC and RC and nowhere else.
4.	`a:b /<= LC _ RC`	A is **never** realized as b in the context LC and RC.

The rules based on these rule types are used to generate a finite state acceptor which executes all rules in parallel and accepts or rejects a lexical-surface pair. The proper sequencing of morphemes (morphotactics) is done by finite state machines that are built by using roots words lexicons and suffixes.

4 Two-Level Rules

We have defined an alphabet for the two-level description of the language. This alphabet includes the standard Turkmen letters and some additional symbols which are used in the intermediate level and have no usage in orthography. We have represent the non-ASCII Turkmen letters by their uppercase counterparts (***ü⇨U, ö⇨O, ç⇨C, ñ⇨N, ş⇨S, ý⇨ Y, ź⇨Z, ä⇨E***). The definitions are listed as the following:

Consonants:	CONS = b C d f g h j Z k l m n N p r s S t w Y z
Vowels :	VOWEL = a E e y i o O u U;
Back Vowels:	BACKV = a y u o
Front Vowels :	FRONTV = e E i O U

Back-Rounded Vowels : BKROV = o u
Back-Unrounded Vowels : BKUNROV = a y
Front-Rounded Vowels : FRROV = O U
Back-Unrounded Vowels : FRUNROV = e i E
First letters of some suffixes which may disappear in some cases: X = s n
Vowels subject to ellipsis under some conditions : VS = y i O o U u

In order to handle phonetic variations, we represent a number of two-level rules taken from various Turkmen language references [10,11,12]. Some important rules are given below:

```
1.  A:a  =>  [:BACKV] [:CONS]* (%':%') (%+:0) [CONS: | :CONS | :0]* _
2.  A:e  =>  [:FRONTV] [:CONS]* (%':%') (%+:0) [CONS: | :CONS | :0]* _
3.  V:a  =>  [:BACKV] [CONS]* (%+:0) [CONS: | :CONS | :0]* _
4.  V:E  =>  [:FRONTV] [CONS]* (%+:0) [CONS: | :CONS | :0]* _
5.  I:y  =>  [:BACKV] [CONS]* (%':%') (%+:0) [CONS: | :CONS | :0]* _
6.  I:i  =>  [:FRONTV] [CONS]* (%':%') (%+:0) [CONS: | :CONS | :0]* _
7.  H:i  =>  [:FRUNROV] [CONS]* (%':%') (%+:0) [CONS: | :CONS | :0]* _
8.  H:y  =>  [:BKUNROV] [CONS]* (%':%') (%+:0) [CONS: | :CONS | :0]* _
9.  H:u  =>  [:BKROV] [CONS]* (%':%') (%+:0) [CONS: | :CONS | :0]* _
10. H:U  =>  [:FRROV] [CONS]* (%':%') (%+:0) [CONS: | :CONS | :0]* _
```

These rules are for the harmony rules of the vowels. The surface realization of ***A***, ***V*** or ***I*** is determined by the backness property of the preceding vowel while an ***H*** is determined by the backness and roundness properties of the preceding vowel.

```
11. H:0  <=>   [:VOWEL] %+:0 _
```

A vowel in ***H*** set is deleted if it is the first letter of a suffix which is affixed to a word (or the previous suffix) which has a vowel as the last letter.

```
12. T:a <=> [:BACKV] [CONS: | :CONS | :0]+ [%+:0] _
13. T:e <=> [:FRONTV] [CONS: | :CONS | :0]+ [%+:0] _
14. Cx:Cy <=> _%+:0 [T:] where Cx in (i e A y) Cy in (E E E a) matched
```

Dative is a non-standard nominal case in Turkmen language. These rules are used to handle these special cases. For example, the words ending with vowels ***i, e*** and ***ä***, these vowels are changed into ***E***.

Lexical :	`Berdi+T`	`(A City Name)+Dative`
Surface:	`BerdE00`	`Berdä`

The vowel ***y*** placed as the last letter of a word is changed into ***a*** in dative case.

Lexical :	`Mary+T`	`(A City Name)+Dative`
Surface:	`Mar00a`	`Mara`

There is no orthographic difference between the nominal case and the dative case for the words ending with the letters ***a*** and ***o***. The diffence is stressed by the duration of the last phoneme in the speech.

Lexical :	`ata+T`	`Noun(father)+Dative`
Surface:	`ata00`	`ata`

```
15. e:E <=>  _ %+:0 [H:] [m: | N: | p:] (I: z:);    _ %+:0 [H:] b e r;
```

This rule changes the letter ***e*** to ***E*** for some cases.

Lexical :	iSle+HpdI	to work+Past
Surface:	iSlE00pdi	işläpdi

```
16. Cx:Cy <=>  [:CONS] %+:0 _ (CONS VOWEL); where Cx in (s n S Y) Cy in
                                                    (0 0 s 0) matched;
```

This rule deletes the letters ***s, n, S*** and ***Y*** if they are in the beginning of a suffix which is affixed to a morpheme that has a consonant as its last letter.

```
17. A:E => %+:0 m _ [k:] %+:0 [T:]; %+:0 d _ %+:0 k [I:]
```

In two special conditions, the ***A*** is resolved as ***E***. One of these conditions is the cases where a verbal root has ***+mAk*** infinitive root and dative case suffixes. Also in a noun which has ***+dA*** locative case and ***+kI*** relative suffixes, the ***A*** of the locative case morpheme is resolved into an ***E***.

Lexical :	ber+mAk+T	to give+Infinitive+Dative
Surface:	ber0mEg0e	bermäge
Lexical :	galam+dA+kI	pencil+Loc+Relative
Surface:	galam0dE0ki	galamdäki

```
18. Cx:Cy => _ %+:0 (X:0) [ :VOWEL ]; where Cx in (p t C k)
                                          Cy in (b d j g) matched;
```

This rule realizes the voiced obstruents ***p***, ***t***, ***C***, and ***k*** when they are followed by a suffix beginning with a vowel.

Lexical :	kitap+T	book+Dative
Surface:	kitab0a	kitaba

```
19. VS:0 <=> %$:0 _  [ CONS %+:0 (X:0) [A: | H: | I: | T:] ];
20. H:0 <=> %$:0 _  [ CONS %+:0 (X:0) [A: | H: | I: | T:] ];
```

In certain words, a vowel ellipsis can occur with some kinds of suffixes. The vowel subject to ellipsis is represented by a preceding ***$*** sign in the lexicon.

5 Morphotactics

The study and modeling of legal word formation is called morphotactic [13]. Morphotactic rules imply the legal ordering of the morphemes. In our implementation, morphotactics are done by finite-state-machines. These machines are depicted in Figure 1 and Figure 2. In these figures, the boxes show the states, the arrows shows the next states that can be reached when a suffix matching one of the labels is found. The circles are the final states which indicate legal word formations. The class of the final word is given in the parentheses beside the final states. The 0 transitions indicate that the transition can be done by the null input. The XFST environment has a module

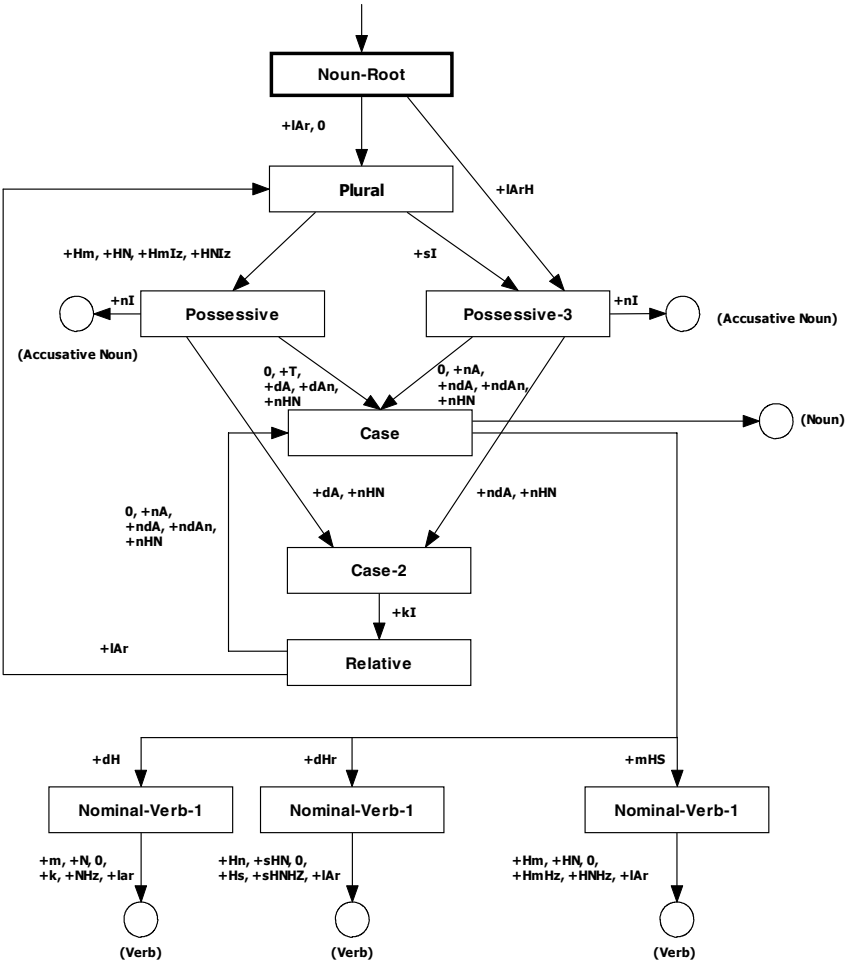

Fig. 1. Finite State Machine for Nominal Morphotactics

called LEXC to build the finite-state-machines as morphotactic rules. A small section of the LEXC lexicons are given below:

```
LEXICON VERBS        LEXICON VERB-POST        LEXICON VERB-ROOT
diY   VERB-POST;     +Verb : 0 VERB-ROOT;     0      : 0     VERB-PASSIVE;
dEl   VERB-POST;                              +Recip : +nHS  VERB-RECIP;
bil   VERB-POST;                              +Recip : +S    VERB-RECIP;
al    VERB-POST;
geple VERB-POST;
```

Each sub-lexicon consists of entries which denote output and input pairs and the name of the next lexicon (state). The system moves to the next state by consuming the input and producing the corresponding output.

Some morphological analysis examples are:

```
galam   0lar        0ym     00yñ    (surface level - galamlarymyñ)
galam   +lAr        +Hm     +nHN    (intermediate level)
pencil  +A3pl       +P1sg   +Gen    (lexical Level - of my pencils)

geple   +mA    +yVr     +Hs   (surface level - geplemeyäris)
geple   0me    0yär     0is   (intermediate level)
talk    +Neg   +Prog1   +A1pl (lexical level - we are not talking)
```

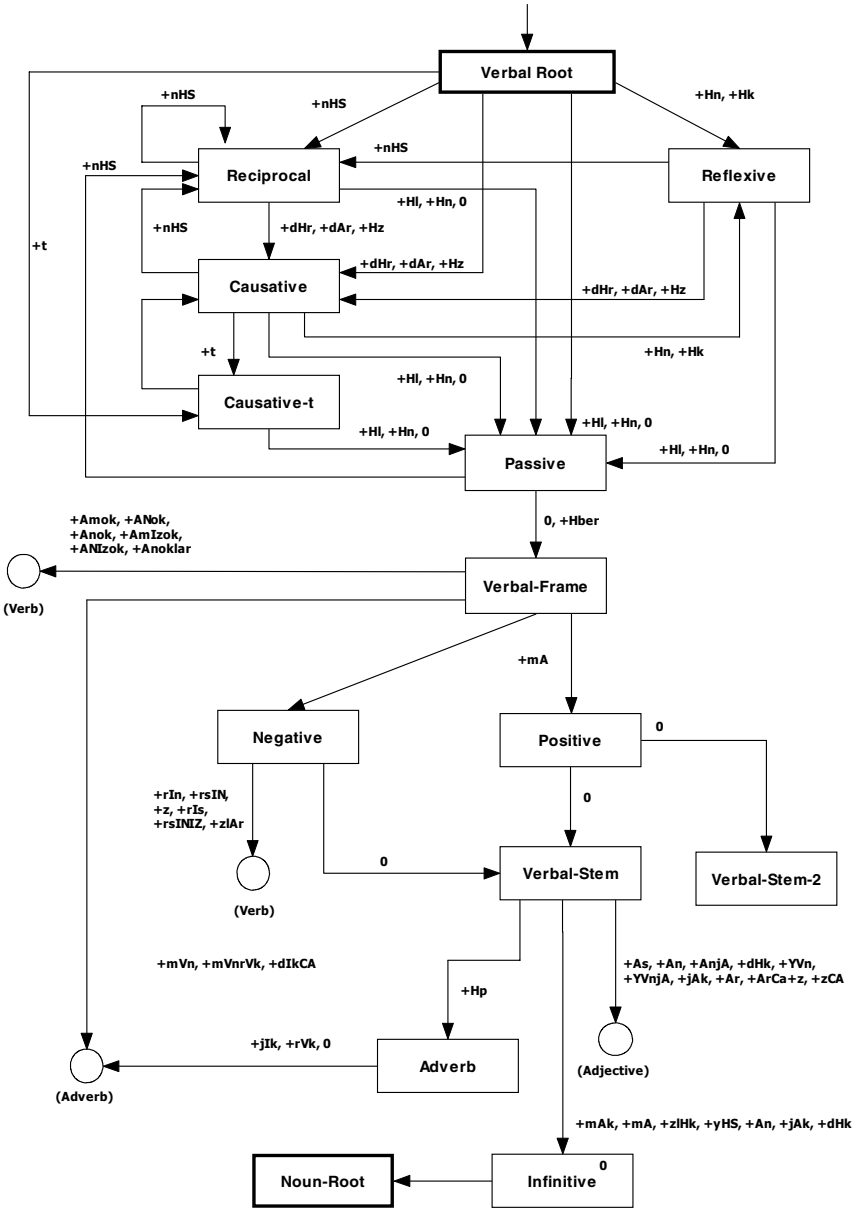

Fig. 2. Finite State Machine for Verbal Morphotactics

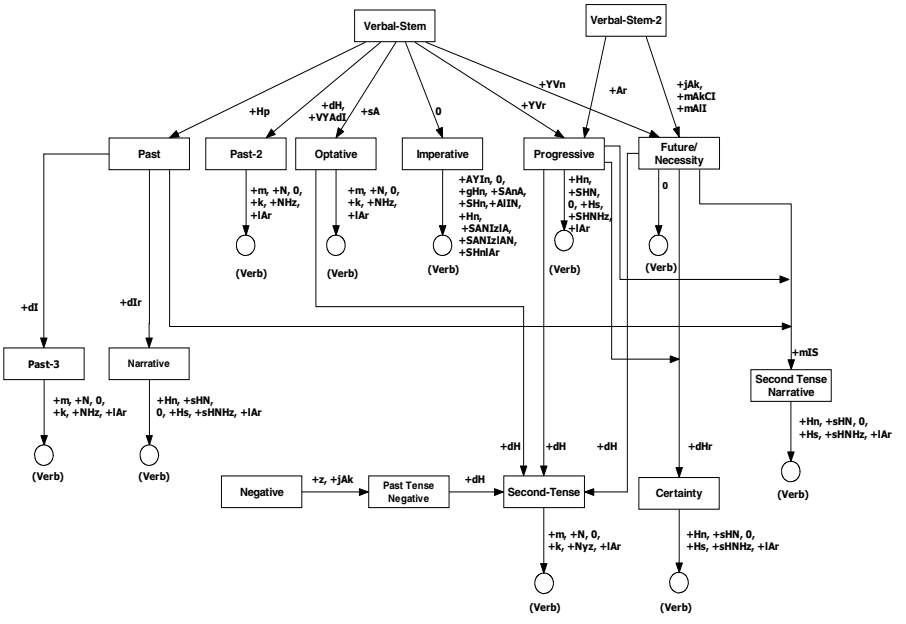

Fig. 2. (*continued*)

6 Conclusion

As a consequence, this work introduces a computer analysis of the Turkmen Language morphology. The well-known two-level morphological analysis method is implemented by finite-state machines. The resulting morphological analyzer is the first step for all kind of NLP tasks because, like Turkish, the Turkmen language has a very complicated inflectional and derivational structure and no other NLP related system can be designed without having the morphological analysis of the words in hand. Even though current version of the implementation does not have large word root lexicons (~1200), it can be easily used for all other NLP related purposes. Since the main goal of implementing a machine translation system between Turkmen and Turkish, we are still improving the performance of the analyzer and enlarging its lexicon size by adding new word roots.

References

1. Sproat, R. : Morphology and Computation, MIT Press (1992)
2. Alam, Y. S. : A Two-Level Morphological Analysis of Japanese. Texas Linguistics Forum, 22:229-252 (1983)
3. Karttunen, L., Wittenburg, K. : A Two-Level Morphological Analysis of English. Texas Linguistics Forum, 22:217-228 (1983)
4. Koskenniemi, K. : An Application of the Two-Level Model to Finnish. In Fred Karlsson, editor, Computational Morphosyntax, a report on research 1981-1984. University of Helsinki Department of General Linguistics (1985)

5. Oflazer, K. : Two-Level Description of Turkish Morphology. Literary and Linguistic Computing, Vol. 9, No:2 (1994)
6. Altintas, K. Cicekli, İ.: A Morphological Analyser for Crimean Tatar. Proceedings of the 10th Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN2001), North Cyprus, pp: 180-189 (2001)
7. Karttunen, L., Gaal, T., Kempe, A. : Xerox Finite-State Tool. Technical Report, Xerox Research Centre, Europe (1997)
8. www.sil.org, www.ethnologue.com
9. Koskenniemi, K. : Two-Level Morphology : A General Computational Model for Word Form Recognition and Production. Publication No:11 , Department of General Linguistics, University of Helsinki
10. Clark, L. : The Turkmen Reference Harrassowitx Verlag, Wiesbaden (1998)
11. Sarı, B., Güder N. : Türkmencenin Grameri (II Morfologiya), Türk Dünyası Gençlerinin Mahtumkulu Yayın Birliği . (1998)
12. Söyegowyñ, M. : Türkmen Diliniñ Grammatikasy – Morfologiya, TDK (2000)
13. Kenneth, B.R., Karttunen, L. : Finite State Morphology, CSLI Publications (2003)