An adaptable morphological parser for agglutinative languages

Marina Ermolaeva
Lomonosov Moscow State University
marinkaermolaeva@gmail.com

Abstract

The paper reports the state of the ongoing work on creating an adaptable morphological parser for various agglutinative languages. A hybrid approach involving methods typically used for non-agglutinative languages is proposed. We explain the design of a working prototype for inflectional nominal morphology and demonstrate its work with an implementation for Turkish language. An additional experiment of adapting the parser to Buryat (Mongolic family) is discussed.[1]

1 Introduction

The most obvious way to perform morphological parsing is to make a list of all possible morphological variants of each word. This method has been successfully used for non-agglutinative languages, e.g. (Segalovich 2003) for Russian, Polish and English.

Agglutinative languages pose a much more complex task, since the number of possible forms of a single word is theoretically infinite (Jurafsky and Martin 2000). Parsing languages like Turkish often involves designing complicated finite-state machines where each transition corresponds to a single affix (Hankamer 1986; Eryiğit and Adalı 2004; Çöltekin 2010; Sak et al. 2009; Şahin et al. 2013). While these systems can perform extremely well, a considerable redesigning of the whole system is required in order to implement a new language or to take care of a few more affixes.

The proposed approach combines both methods mentioned above. A simple finite-state machine allows to split up the set of possible affixes, producing a finite and relatively small set of sequences that can be easily stored in a dictionary.

Most systems created for parsing agglutinative languages, starting with (Hankamer 1986) and (Oflazer 1994), process words from left to right: first stem candidates are found in a lexicon, then the remaining part is analyzed. The system presented in this paper applies the right-to-left method (cf. (Eryiğit and Adalı 2004)): affixes are found in the first place. It can ultimately work without a lexicon, in which case the remaining part of the word is assumed to be the stem; to improve precision of parsing, it is possible to compare it to stems contained in a lexicon. A major advantage of right-to-left parsing is the ability to process words with unknown stems without additional computations.

Multi-language systems (Akın and Akın 2007; Arkhangelskiy 2012) are a relatively new tendency. With the hybrid approach mentioned above, the proposed system fits within this trend. As the research is still in progress, the working prototype of the parser (written in Python language) is currently restricted to nominal inflectional morphology. Within this scope, it has been implemented for Turkish; an additional experiment with Buryat language is discussed in the section 5.

2 Turkish challenges

The complexity of Turkish morphology is easily perceptible in nouns. The word stem itself can be complex. Compounding of “adjective + noun” or “noun + noun” structure is a productive way of word formation, which means that this problem cannot be solved by listing all known compounds in a dictionary.

[1] This is a slightly altered version of the following paper:
Marina Ermolaeva. 2014. An adaptable morphological parser for agglutinative languages. In: Proceedings of the First Italian Conference on Computational Linguistics CLiC-it 2014 & the Fourth International Workshop EVALITA 2014. Vol. I. Roberto Basili, Alessandro Lenci, Bernardo Magnini (eds.). pp. 164-168. Pisa University Press.

Due to the vowel harmony and assimilation rules, most affixes have multiple allomorphs distributed complementarily according to the phonological context; e.g. the locative case marker has 4 forms (two harmonic variants of the vowel and a voiced/voiceless alternation).

A nominal stem can receive number, possession and case affixes. Moreover, certain other affixes (e.g. copular and person markers) can attach to these forms to form predicates:

(1) ev-ler-imiz-de-ymiş-ler[2]
home-PL-P1PL-LOC-COP.EV-3PL
Apparently they are/were at our homes.

An interesting option is the affix *-ki*, which can be recursively attached to a nominal form containing a genitive or locative marker:

(2) ev-de-ki-ler-in-ki
home-LOC-KI1-PL-GEN-KI2[3]
the one belonging to those at home

3 System design

3.1 Data representation

The language-specific data necessary to implement a new language includes:

- Phonology description (phoneme inventory, harmony, etc.)
- Morphology description: a list of all allomorphs. For each allomorph its category, gloss and possible (morpho)phonological context is stored.
- Lexicon: a list of stems with part-of-speech tags. If a stem has multiple phonological variants, they are stored as separate entries along with data about contexts they can be used in. The lexicon is optional, yet it significantly improves precision of parsing.

The parser itself is language-independent and does not require any custom coding to implement new languages.

For Turkish, the system uses a relatively small lexicon of 16000 nominal and adjectival stems. The modest size of the lexicon is mostly compensated by the ability to analyze morphology even if the stem is absent in the lexicon. In this case, parses for all possible stems are output.

The exceedingly long morpheme sequences that can attach to a stem are split up into shorter chains. The whole set of grammatical categories is represented as a set of slots, each of them containing categories that have strictly fixed order(s):

- two stem slots (for nominal compounds)
- noun inflection
- noun loop (the recursive suffix *-ki*)
- nominal verb suffixes (e.g. copulas and adverbial markers)

The number and order of categories within slots can be changed without modifying the system itself, which simplifies implementing new languages.

For each slot, a list of possible affix sequences is obtained. At this step all the checks of morphotactic and phonological compatibility of the affixes within a slot are performed, so they do not have to be applied at runtime. The lists are converted into tries in order to speed up the search. All the sequences are stored inverted, so that the trie could be searched during the parsing process. A fragment of the nominal morphology trie and the sequences compatible with it are shown in Figure 1 and Table 1 respectively.

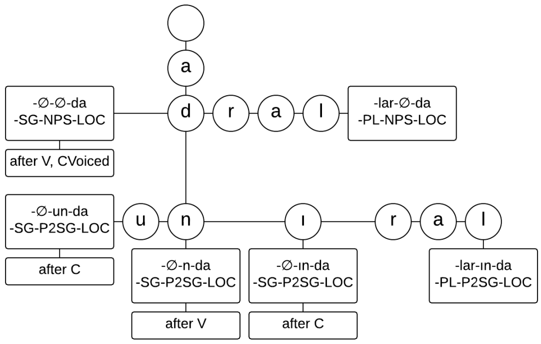

Figure 1. A fragment of the nominal affix trie

Sequence	Gloss	Context
-∅-∅-da	-SG-NPS-LOC	after vowels and voiced consonants
-∅-un-da	-SG-P2SG-LOC	after consonants
-∅-n-da	-SG-P2SG-LOC	after vowels
-lar-ın-da	-PL-P2SG-LOC	(no restrictions)
-∅-ın-da	-SG-P2SG-LOC	after consonants
-lar-∅-da	-PL-NPS-LOC	(no restrictions)

Table 1. Sequence list for Figure 1

Similarly, the lexicon is stored as a set of tries. Stems are also inverted, in order to effectively find stem boundaries within compounds. Stems with multiple phonological variants are included in the lexicon as a set of

[2] Examples (1)-(4) are from (Göksel and Kerslake 2005)

[3] According to Hankamer (2004), *-ki* has different properties when attached to a locative form and to a genitive form; therefore, two separate *-ki*'s are postulated. In this paper, they are referred to as KI1 and KI2 respectively.

separate entries; each entry receives special labels determining possible phonological context. For instance, *his* “sensation” appears in the form *hiss* before vowels and in the vocabulary form in other cases. A fragment of the lexicon trie is represented in Figure 2; it corresponds to the list of stems in Table 2.

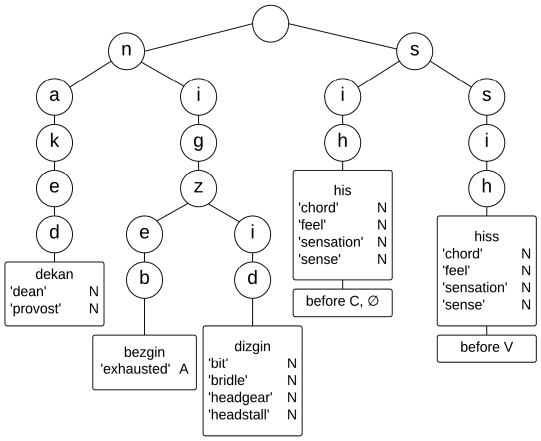

Figure 2. A fragment of the lexicon trie

Sequence	Translation(s)	Context
dekan	dean, provost	(no restrictions)
bezgin	exhausted	(no restrictions)
dizgin	bit, bridle, …	(no restrictions)
his	chord, feel, …	before consonants; at the word’s end
hiss	chord, feel, …	before vowels

Table 2. Sequence list for Figure 2

3.2 Parsing algorithm

The transitions between slots are performed via a (very simple) finite-state machine shown in Figure 3:

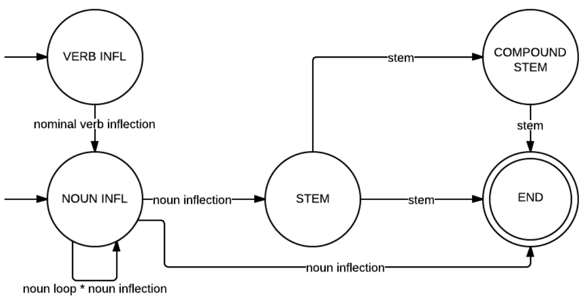

Figure 3. The finite-state machine

Each transition corresponds to a sequence of affixes rather than to a single affix. Each transition involves finding all possible candidate sequences using an appropriate stem or affix trie. Checks of compatibility are only done between slot sequences; at other points, no linguistic information is used. The simplified algorithm of analysis includes following steps:

1. Find all affix sequences that match the input word form.
2. For each hypothetical parse, try to find a stem in the lexicon using the unglossed part at the word’s left end. If a stem is found and there are no “leftover” characters at the left end of the word, output all such parses. If a stem is found, yet some part of the word remains unglossed, go to step 3. If no stem is found at all, assume that the stem is unknown and output all hypothetical parses.
3. Assume that the stem is compound; for the remaining unglossed part, try to find another stem. If a stem is found and no unprocessed characters are left, output all such parses. Else discard the hypothetical compound parses and output all parses with no stem found.

Some examples of different decisions made by the algorithm are demonstrated below. In (3), the input is ambiguous. For two of the possible stem-affix boundaries (*adam-dı* and *ada-mdı*), a known stem has been found in the lexicon:

(3) **input**: *adamdı*
decision: single stem
output:
1. adam-∅-∅-∅-dı-∅
man-SG-NPS-NOM-COP.PST-3
2. ada-∅-m-∅-dı-∅
island-SG-P1SG-NOM-COP.PST-3

Even if there is no single stem matching the input in the lexicon, like in (4), a suitable parse might be found under the assumption that there is an additional boundary within the stem:

(4) **input**: *kızarkadaş*
decision: compound
output:
1. kız-arkadaş-∅-∅-∅
girl-friend-SG-NPS-NOM
2. kız-arkadaş-∅-∅-∅-∅-∅
girl-friend-SG-NPS-NOM-COP.PRS-3

Finally, the pseudo-word in (5) has two feasible stem-affix boundaries (with hypothetical stems *fefe* and *fef*), but no single or compound match in the lexicon for any of them. The stem is considered unknown, and all parses are output:

(5) **input**: *fefe*
decision: unknown stem
output:
1. fef-∅-∅-e
FEF-SG-NPS-DAT
2. fef-∅-∅-e-∅-∅
FEF-SG-NPS-DAT-COP.PRS-3

3. fefe-∅-∅-∅
   FEF-SG-NPS-NOM
4. fefe-∅-∅-∅-∅-∅
   FEF-SG-NPS-NOM-COP.PRS-3

4 Evaluation

Turkish is known for a significant level of morphological ambiguity. For example, it is impossible to disambiguate (6) and (7) without appealing to the context:

(6) ev-in
house-GEN
'of the house'

(7) ev-in
house-P2SG
'your house'

Since the system does not perform disambiguation, it must output all possible parses for each word. To take this into account, the evaluation method described in (Paroubek 2007) has been used. First, precision (P) and recall (R) values for each word w_i in the test sample are obtained:

$P(w_i) = \frac{t_i \cap r_i}{t_i}$; $R(w_i) = \frac{t_i \cap r_i}{r_i}$,

where t_i is the number of parses for w_i output by the parser and r_i is the number of correct parses.

After that, mean values for the whole sample are calculated. As most derivational affixes are currently not regarded, the internal structure of the stem was not considered. A parse was accepted if all inflectional affixes had been correctly found and properly labelled.

The Turkish implementation was evaluated with a testing sample of 300 nouns and noun-based predicates and yielded precision and recall values of 94,8% and 96,2% respectively.

5 Implementing new languages

Since Turkic languages are quite similar among themselves, applying the parser to a non-Turkic agglutinative language can help test its universality.

As an experiment, a small part of Buryat morphology has been modelled. Buryat language poses more challenges than Turkish in some respects. The processing is complicated by a vast number of (morpho)phonological variants of both stems and affixes, more complex phonological rules and a harmony system with subtler distinctions (e.g. a distinction between vowels in different syllables).

Crucially, the Buryat implementation did not require any custom coding or language-specific modifications of the parser itself; the only custom elements were phonology description, morpheme list and dictionary. The morphology model was evaluated on a small sample of Buryat nouns, resulting in precision value of approximately 91% and recall value of 96%.

6 Future work

At the moment, the top-importance task is lifting the temporary limitations of the parser by implementing other parts of speech (finite and non-finite verb forms, pronouns, postpositions etc.) and derivational suffixes.

Although the slot system described in 3.1 has been sufficient for both Turkish and Buryat, other agglutinative languages may require more flexibility. This can be achieved either by adding more slots (thus making the slot system nearly universal) or by providing a way to derive the slot system automatically, from plain text or a corpus of tagged texts; the latter solution would also considerably reduce the amount of work that has to be done manually.

Another direction of future work involves integrating the parser into a more complex system. DIRETRA, an engine for Turkish-to-English direct translation, is being developed on the base of the parser (Aksënova and Ermolaeva in prep.). The primary goal is to provide a word-for-word translation of a given text, reflecting the morphological phenomena of the source language as precisely as possible. The gloss lines output by the parser are processed by the other modules of the system and ultimately transformed into text representations in the target language:

input	adamlarınkiler
parser output	man-PL-GEN-KI2-PL
DIRETRA output	ones.owned.by.men

Table 3. An example of DIRETRA output

Though the system is being designed for Turkish, the next step planned is to implement other Turkic languages as well.

Abbreviations

1 – first person, 2 – second person, 3 – third person, COP.EV – evidential copula, COP.PRS – present tense copula, COP.PST – past tense copula, DAT – dative, GEN – genitive, KI1 – -ki suffix after locative, KI2 – -ki suffix after genitive, LOC – locative, NOM

– nominative, NPS – non-possession, P – possession, PL – plural, SG – singular.

References

Ahmet Afşın Akın and Mehmet Dündar Akın. 2007. Zemberek, an open source NLP framework for Turkic Languages.

Alëna Aksënova and Marina Ermolaeva. In prep. DIRETRA, a customizable direct translation system: first sketches. In: Proceedings of Translata II.

Timofey Arkhangelskiy. 2012. Printsipy postrojenija morfologicheskogo parsera dlja raznostrukturnyx jazykov [Principles of building a morphological parser for different-structure languages] Abstract of thesis cand. phil. sci. Moscow.

Çağrı Çöltekin. 2010. A Freely Available Morphological Analyzer for Turkish. In: Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC2010), Valletta, Malta.

Gülşen Eryiğit and Eşref Adalı. 2004. An Affix Stripping Morphological Analyzer for Turkish. In: IASTED International Multi-Conference on Artificial Intelligence and Applications. Innsbruck, Austria, 299-304.

Aslı Göksel and Celia Kerslake. 2005. Turkish: A Comprehensive Grammar.

Jorge Hankamer. 1986. Finite state morphology and left-to-right phonology. In: Proceedings of the Fifth West Coast Conference on Formal Linguistics, Stanford, CA, 29-34.

Jorge Hankamer. 2004. Why there are two ki's in Turkish. In: Imer and Dogan, eds., Current Research in Turkish Linguistics, Eastern Mediterranean University Press, 13-25.

Daniel Jurafsky and James H. Martin. 2000. Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition. Upper Saddle River, N.J.: Prentice Hall.

Kemal Oflazer. 1994. Two-level description of Turkish morphology. In Literary and Linguistic Computing, vol. 9, no. 2, 137-148.

Patrick Paroubek. 2007. Chapter 4 - Evaluating Part Of Speech Tagging and Parsing. In: Evaluation of Text and Speech Systems, eds. Laila Dybkjær, Holmer Hemsen, Wolfgang Minker, series: Text, Speech and Language Technology, vol. 36, Kluwer Academic Publisher, 97-116.

Muhammet Şahin, Umut Sulubacak and Gülsen Eryigit. 2013. Redefinition Of Turkish Morphology Using Flag Diacritics. In: Proceedings of the Tenth Symposium on Natural Language Processing (SNLP-2013).

Haşim Sak, Tunga Güngör and Murat Saraçlar. 2009. A stochastic finite-state morphological parser for Turkish. In: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, August 04-04, 2009, Suntec, Singapore.

Ilya Segalovich. 2003. A Fast Morphological Algorithm with Unknown Word Guessing Induced by a Dictionary for a Web Search Engine. MLMTA, 273-280. CSREA Press.