The Third International Conference "Problems of Cybernetics and Informatics"
September 6-8, 2010, Baku, Azerbaijan. Section #1 "Information and Communication Technologies"
www.pci2010.science.az/1/28.pdf

SOFTWARE FOR PROCESSING OF NATURAL LANGUAGE TEXTS

Jemal Antidze[1], Nana Gulua[2]

[1]Tbilisi State University, Vekua Scientific Institute of Applied Mathematics, Tbilisi, Georgia,
[2]Nana Gulua, Sokhumi State University, Tbilisi, Georgia
[1]*jeantidze@yahoo.com*, [2]*ngulua7@mail.ru*

The computer morphological analysis of Georgian words is one of the main components for solving such problems as machine translation from Georgian language to the other languages, as well as the automated checking of orthography of Georgian texts, and some problems of artificial intelligence, which require computer processing of Georgian texts. The complete system for computer morphological analysis of Georgian words does not exist yet. If we need to use Georgian language to communicate with computer, the solving of mentioned above problem is very urgent.

For solving this problem using of finite automaton, which is widely used for the languages from Western Europe, is not feasible. This is happening because of some verb-forms of Georgian language require backtracking, which is impossible with finite automaton. From the other side, using of full search algorithm slows the process of morphological analysis. For this reason, we formed a method, which is making the analysis process faster, compare to full search algorithm [1]. This method uses constraints to establish correct morpheme's selection. Already separated presumable morphemes from word, morphological analysis tool checks it on satisfaction of their constraints. If the constraint is satisfied, the tool continues separation of other morphemes in opposite case it performs backtracking to search the new alternatives and rejects the last separated morpheme. In this way, the process of removing of incorrect alternatives happens in advance, what speeds up the searching process. The constraints are logical expressions, which we can compose from the features of morphemes. The tool checks, if separated morpheme's feature has particular value, which defines correctness of the separation. We compose the values of morphemes' features according to morphology of Georgian language.

Under complete computer morphological analysis, we understand all valid splitting of a word-form in morphemes and establishment of morphological categories for each splitting. The definition contains ambiguities of words. The following ambiguity is widespread:

1. Graphical coincidence of different verb-forms (by meaning) in presence circle, which have the same root. For instance, verb-form "agebs", which may mean loss (many**)** or build (plan) and so on;
2. Graphical coincidence of a verb-form with its infinitive, for instance, "amoxsna" may mean, "resolution" or "he has resolved";
3. In time of splitting of verb-form, graphical coincidence of morphemes from different neighboring classes, for instance, "a" as the preverbal or vowel prefix or first letter of a verb's root in the following verb-forms: "a-a-alebs", "a-alebs" and "aldeba". When we see first letter of the verb-form "aaalebs", we cannot say, which morpheme we have, before we have seen following two letters. In first example, first "a" is preverbal. In second example, first "a" is vowel prefix and in third example first "a" is first letter of the root "al". This means, that Georgian verbs splitting in morphemes needs at least parsing algorithm for LL(2) grammar ([2]), i.e. complete morphological analysis of Georgian words by finite automaton is impossible.

In the second case, morphological analysis for verb-form "amoxsna" must give two different parsing: one- for infinitive and second - for verb-form. For this, we need nondeterministic algorithm. Deterministic algorithm cannot give two different parses for the

same word-form. Thus, deterministic algorithm is not valid for complete morphological analysis of Georgian words. All author fulfilled morphological analyses for Georgian words by finite automaton or by deterministic algorithm [3, 4]. For complete morphological analysis, we must apply non-deterministic algorithm, for instance, from left to right in depth search algorithm with backtrackings. As far as backtrackings take down the speed of the algorithm, we must find a method, which reduces them. Such possibility exists. We can exclude morphemes, which conflict with found morphemes at a moment. In other case, we can divide morphemes in classes so, that one representative of each class will meet as maximum one times in a word-form. Among morphemes of a verb-form are important roots. We can divide roots into classes so, that each morpheme, which can meet in a word-form, will indicate definitely a morphological category. All this reduces backtrackings and establishing morphological categories considerably. After this, the establishment of morphological categories of a word is easy. We realized complete morphological analysis of Georgian words by the tool [5-7].

The Software is designed for the processing of natural language texts. We use the system to nalyze syntactic and morphological structure of the natural language texts. Using specific formalism, which we created for this purpose, allow us to write down syntactic and morphological rules defined by particular natural language grammar. This formalism represents the new, complex approach, which solves problems of morphological and syntactic analysis for some natural language. We implemented a software system according to this formalism [1]. One can realize syntactic analysis of sentences and morphological analysis of word-forms with this software system. We designed several special algorithms for this system. Using the formalism, which is described in [8, 9], is very difficult to use for Georgian language, as far as expressing of some morphological rules is very complicated and understanding of such writing is difficult.

The software consists of two parts: syntactic analyzer and morphological analyzer. Purpose of the syntactic analyzer is to parse an input sentence, to build a parsing tree, which describes relations between the individual words within the sentence, and to collect information about the input sentence, which the system figured out during the analysis process. It is necessary to provide a grammar file to the syntactic and morphological analyzers. There must be recorded syntactic or morphological rules of particular natural language grammar. Basic methods and algorithms, which we used to develop the system, are operations defined on features' structures; trace back algorithm (for morphological analyzer); general syntactic parsing algorithm for context free grammar and features' constraints method. Features' structures are widely used on all levels of analysis. We use them to hold various information about dictionary entries and information obtained during analysis. Each symbol defined in a morphological or syntactic rule has an associated features' structure, which we initially fill from the dictionary, or the system fill them by the previous levels of analysis. Features' structures and operations defined on them we use to build up features' constraints. With general parsing algorithm, it is possible to get a syntactic analysis of any sentence defined by a context free grammar and simultaneously check features' constraints, which may be associated with grammatical rules. Features' constraints are logical expressions composed by the operations, which we defined on the features' structures. We attach features' constraints to rules, which we defined within a grammar file. If the constraint is not satisfied during the analysis, then the system will reject current rule and the search process will go on. We can attach features' constraints also to morphological rules. However, unlike the syntactic rules, we can attach constraints at any place within a morphological rule, only not at the end. This speeds up morphological analysis, because the system checks constraints early and it rejects incorrect word-form's division into morphemes in a timely manner.

Formalism, which we developed for the syntactic and morphological analysis is highly comfortable for human. It has many constructions that make it easier to write grammar file.

Morphological analyzer has a built-in preprocessor. It utilizes STL standard library. Program operates in UNIX and Windows operating systems. We can compile it and use in any other platform, which contains modern C++ compiler.

In our system, we use features' structures and operations defined on them to put constraints on parser rules. That makes parser rules more suitable for natural language analysis than pure CFG rules. We have generalized notation of constraint [2]. Constraint is any logical expression built up with operations defined on features' structures and basic logical operations and constants: & (and), | (or), ~ (not), 0 (false), 1 (true).

Parser rules we can write following way:

$$S->A1\{C1\}A2\{C2\}...An\{Cn\}$$

Where *S* is an LHS non-terminal symbol, *Ai* (*I* = 1, …, *N*) are terminal or non-terminal symbols (for morphological analyzer only terminal symbols are allowed), and *Ci*(*I* = 1, … , *N*) are constraints. Each constraint is check as soon as all of the RHS symbols located before we match the constraint to the input. If a constraint evaluates to "true" value then parser will continue matching, otherwise if constraint evaluates to "false" parser will reject this alternative and will try another alternative. There is a features' structure associated with each (S and A_i) symbol in a rule. If a symbol is a terminal symbol, then we take initial content of its associated features' structure from the dictionary or from the morphological analyzer (for syntactic analyzer). We take content for a non-terminal symbols from the previous levels of analysis. We use constraints not only to check the correctness of parsing and not only to reduce unnecessary variants. We also use them to transfer data to a LHS symbol, thus move all necessary information to the next level of analysis. We can use assignment or unification operations for this purpose. To access a features' structure for particular symbol, we can use a path notation. We write a path using angle brackets. For example, <A> represents a features' structure associated with the A symbol. We can access individual fields by listing all path components in angle brackets.

Purpose of morphological analyzer is to split an input word into the morphemes and figure out grammar categories of the word. We may invoke morphological analyzer manually or automatically by the syntactic analyzer.

We used special formalism to describe morphology of natural language and pass it to the morphological analyzer. There are two main constructions in the grammar file of morphological analyzer: morphemes' class definition, and morphological rules [10]. Morphemes' class definition is used to list all possible morphemes for a given morphemes' class. For example:

$$@M = \{"morpheme\_1"[...features...]"morpheme\_2"[...features...]...$$

$$"morpheme\_n"[...features...]\}$$

It is possible to declare empty morpheme, which means that we may omit the morphemes' class in morphological rules. Below is formal syntax for morphemes' class definition:

$$<morpheme\_definition>::="@"<identifier>"="{"<list\_of\_morphemes>"}"$$

$$<list\_of\_morphemes>::=<morpheme>\{","<morpheme>\}$$

$$<morpheme>::=<string><feature\_structure>...$$

We define morphological rules following way:

$$Word \mapsto M1\{C1\}M2\{C2\}...Mn\{Cn\}$$

Where Mi are morpheme classes, and *Ci* (*i* = 1, … , *n*) are constraints (optional).

Purpose of syntactic analyzer is to analyze sentences of natural language and produce parsing tree and information about the sentence. In order to accomplish this task, syntactic analyzer needs a grammar's file and a dictionary (or it may use morphological analyzer instead of complete dictionary). We write grammar rules for syntactic analyzer like CFG rules.

However, they may have constraints and symbol position regulators. We can write the rule according to these conventions:

$S \mapsto A1\{C1\}A2\{C2\}...An\{Cn\}$

$S \mapsto A1A2...An : R\{C\};$

Where *S* is an LHS non-terminal symbol *Ai* (*I* = 1, …, *n*) are RHS terminal or non-terminal symbols, *C* and *Ci* (*i* = 1, … , *n*) are constraints, and *R* is a set of symbol position regulators. Position regulators declare order of RHS symbols in the rule, consequently making non-fixed word ordering. There are two types of position regulators:

Ai < *Aj* means that symbol *Ai* must be placed somewhere before the symbol *Aj*

Ai - *Aj* means that symbol *Ai* must be placed exactly before the symbol *Aj*

Described software tools we used for morphological and syntactic analyses of Georgian texts. All problems mentioned above were resolved. We simplified composition of grammar file by using macros with parameters.

References

1. J. Antidze, D.Mishelashvili. Software Tools for Morphological and Syntactic Analysis of Natural Language Texts. (In Georgian) Computer Sciences and Telecommunications, 1(12), Tbilisi (2007) 10p. http://gesj.internet-academy.org.ge/gesj_articles/1345.pdf
2. J. Antidze. Theory of Formal Languages and Grammars, Natural Languages Computer Modeling. (In Georgian) "Nekeri", Tbilisi (2009) 350 p.
3. K. Datukishvili, M. Loladze, N.Zakalashvili. Georgian Language Processing (morphological level). (In English) Report of Symposium – Natural Language Processing, Georgian Language and Computer Technologies, Tbilisi (2003) 1 p.
4. L. Margvelani. Machine Analysis System of Georgian Word Forms. (In English) Report of Symposium – Natural Language Processing, Georgian Language and Computer Technologies, Tbilisi (2003) 1 p.
5. J. Antidze, D. Mishelashvili. Instrumental Tool for Morphological Analysis of Some Natural Languages. (In English) Reports of Enlarged Session of the Seminar of IAM TSU, vol.19, Tbilisi (2004) 5p.
6. J. Antidze, D. Melikishvili, D. Mishelashvili. Georgian Language Computer Morphology. (In English) Conference – Natural Language Processing, Georgian Language and Computer Technology, Tbilisi (2004) 1 p.
7. J. Antidze, N. Gulua. On selection of Georgian texts computer analysis formalism. (In English) Bulletin of The Georgian Academy of Sciences, 162, N2, Tbilisi (2000) 4 p.
8. S. McConnell. PC-PATR: Reference Manual, a unification based syntactic parser, version 1.2.2 , (In English) http://www.sil.org/pcpatr/manual/pcpatr.html
9. E. Antworth, S. McConnell. PC-Kimmo Reference Manual, A two-level processor for morphological analysis, version 2.1.0. (In English) http://www.sil.org/pckimmo/v2/pc-kimmo_v2.html
10. D. Melikishvili. The Georgian Verb: A Morphosyntactic Analysis. (In English) Dunwoody Press, New York (2008) 742 p.