Using XML and Regular Expressions in the Syntactic Analysis of Inflectional Language[⋆]

Marek Trabalka and Mária Bieliková

Slovak University of Technology
Department of Computer Science and Engineering
Ilkovičova 3, 812 19 Bratislava, Slovakia
trabalka@dcs.elf.stuba.sk, bielik@elf.stuba.sk

Abstract. In this paper we describe an approach to representation of data and knowledge using two technologies: XML and regular expressions in a domain of natural language syntactic analysis. Analysis of text written in natural language requires several lexicons that aid the process of syntactic analysis. Moreover knowledge about the language (e.g., syntactic rules) should be represented and interpreted. An effective syntactic parser requires effective representation and manipulation of these data and knowledge. XML and regular expressions allow unified representation of static and dynamic aspects of the natural language analysis. One of the major features of our approach is its extensibility and open character. We used this representation within our method of syntactic analysis, which is based on the bottom-up model.

1 Introduction

Syntactic parsing plays an important role in the natural language processing. It can be used for tasks related to correcting documents and searching relevant information. The mentioned area of research has growing importance with the growth of the use of World Wide Web as an environment for sharing information. Syntactic analysis can be employed for improving search on Internet [3, 2]. The analysis allows, for example, extracting noun phrases of the user request written in natural language, which can serve for further searching or filtering relevant information. Analysis of the text can be employed also in the task of classification of documents.

For the tasks mentioned above an efficient syntactic parser is required. We deal with the problem of syntactic analysis for inflectional languages, i.e. languages, where words have usually several different morphological forms that are created by changing a suffix.

Analysis of text written in the inflectional language requires several lexicons, which aid the process of syntactic analysis. Syntactic analysis depends heavily

[⋆] The work reported here was partially supported by Slovak Science Grant Agency, grant No. 95/5195/605 and project INCO Copernicus, No. 977069, European Research Network for Intelligent Support of Electromyographic Studies: EMG-Net.

on our knowledge about the language (e.g., knowledge about paradigms and their forms). An effective syntactic parser requires effective representation and manipulation of these data and knowledge. An effective implementation of the natural language syntactic parser requires that we represent these knowledge and reason effectively about them.

The purpose of this paper is to report on our approach to representation of data and knowledge using the eXtensible Markup Language (XML) and regular expressions. Combination of XML and regular expressions provides interesting and robust features of the application. It enables unified representation of static and dynamic aspects of the natural language analysis. One of the major features of this approach is its extensibility and open character. We describe the use of such representation within our method of syntactic analysis, which is based on the bottom-up model.

2 Syntactic Parser for Inflectional Language

A syntactic parser can be designed and implemented in a few lines of program (in languages, which allow effective representation of the parser, e.g. Prolog or Perl). Unfortunately such "simple" solutions produce exponential growth of possible word combinations during the analysis. Problem arises with the efficiency and then usability of such parser. Most of today's research and practice in syntactic parsing is concerned with the English language. The concentration on English has resulted in advances in solving of its linguistic problems. However parsing text written in other languages often requires solving different problems. Performance of the syntactic parsers for the English language is often not satisfactory, in particular for inflectional languages [12].

We proposed bottom-up syntactic parsing method, which starts with words in a sentence (the terminals) and attempts to find series of reductions that yield the sentence. This approach is suitable for inflectional languages (e.g., Slavic languages such as the Slovak, Czech, or Russian languages), which are characteristic by their relatively high degree of word order freedom. Languages like English have strong word order and the position of the word expresses its role in the sentence. In that situation, usage of a top-down parser is highly recommended because of its prediction features. On the other hand the order of sentence constituents in the inflectional language is not strict and often is driven more by a human intuition rather than by firmly given rules [8]. Therefore design of the top-down syntactic parser for the inflectional language requires consideration of many possible combinations of words in the sentence.

In the inflectional language it is possible to determine role of the word even at the centre of the sentence when no analysis of the previous words has been made. We adopted this feature to parse the sentence in an effective way. We use morphological categories of words and syntactic rules for connecting words into the phrases to built larger and larger phrases and finally combine them into the sentence. In our method, words are not processed from left to right (such

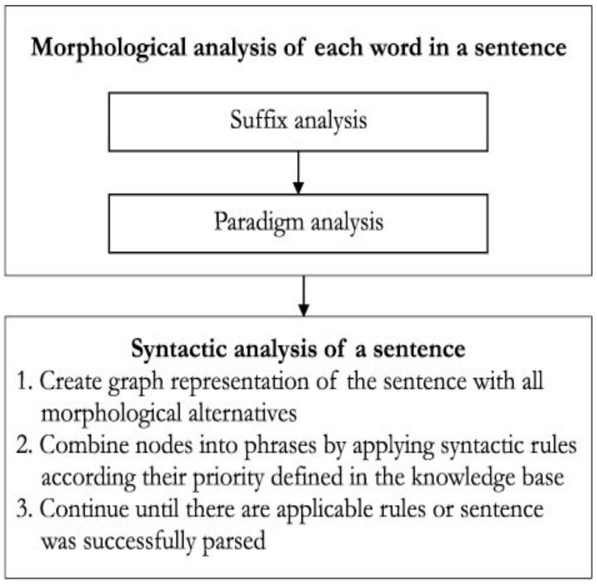
Morphological analysis of each word in a sentence
Suffix analysis
Paradigm analysis
Syntactic analysis of a sentence
1. Create graph representation of the sentence with all morphological alternatives
2. Combine nodes into phrases by applying syntactic rules according their priority defined in the knowledge base
3. Continue until there are applicable rules or sentence was successfully parsed

Fig. 1. Process of a sentence analysis.

as e.g. in [5]), but processing depends on a word's potential role and priority of syntactic rules defined in the knowledge base.

Analysis of a sentence is performed at the three levels: phonological, morphological and syntactical. Phonological analysis is aimed to the sound examination that is combined to form a language. We consider phonology only at the level of symbols recognition, which belong to the particular language. Figure 1 illustrates the main steps of our method of the sentence analysis, which comprises morphological and syntactic levels.

At the beginning of the process all possible morphological categories of each word are found. Our approach to morphological analysis is based on subsequent refining of partial results. A base form of a word is created and its presence is checked in the lexicon. Next, the suffix of the word is analysed in order to filter impossible forms. Finally, all potential paradigms are checked to find matching forms. The result of morphological analysis is often ambiguous. Determined base form and grammatical categories of matching word constitute a basis for syntactic parsing.

The principle of the suffix analysis is based on the fact that many words have typical endings, which determine some of their morphological categories. This approach is applicable also for checking letters before the actual grammatical suffix (which is used obviously in the suffix analysis). For example, the suffix *-pcia* is in the Slovak language typical for famine nouns in singular nominative. However, the suffix *-a* (grammatical suffix of words with the suffix *-pcia*) is much more ambiguous. It can determine verb, noun, or adjective in various numbers cases and genders.

In order to perform suffix analysis we checked a stock of language words and defined meaningful suffixes into a lexicon. The main advantage of the proposed approach is that the process of suffix analysis is straightforward and fast. It can also disambiguate some alternatives in very early stage of the analysis. However, such analysis can produce more results (possible word categories) because not all suffixes are unambiguous. This limitation is resolved by including the adaptive morphological analysis and the syntactic analysis that reduce the number of possible alternatives.

The method of adaptive morphological analysis [10] automates the process of binding an unknown word to an appropriate paradigm. It uses knowledge about paradigms and heuristics to compute an acceptance probability of the form for the particular word. Input to the method is a word, which is subject to the morphological analysis (in an arbitrary form). Output presents determination of the form for the analysed word (a set of morphological tags) with an acceptance probability. Output can be ambiguous, i.e. several different forms with their acceptance probabilities are returned. Alternatives and their acceptance probabilities are computed by means of linguistic knowledge. Linguistic knowledge is either filled in advance by an expert, or learned by the system during the previous analysis.

By the use of morphological analysis an initial graph for syntactic analysis is created. The nodes represent forms of words. Oriented edges connect forms of neighbouring words. The process of syntactical analysis then creates new nodes using syntactic rules about connections forms into a phrase. The new node represents particular phrase. It is connected to all predecessors of the first node and to all successors of the last node of the phrase was made of.

The parser has to remember the application of each rule on each combination of nodes. If a node without neither predecessors nor successors is created, parsing is successful. Such node represents full sentence and parsing trees can be created by analysing history of the analysis. If there is no such node and no syntactic rule can be further applicable, parsing was unsuccessful.

Our approach takes advantage of refined processing of the sentence in several adjoin phases. The analysis exploits the morphological information about particular suffixes. Suffix analysis often produces ambiguous results although in the particular context the word has often only one meaningful form. This limitation is resolved by including the adaptive morphological analysis [10] and the syntactic analysis that starts analysis with the most promising part of the sentence. The number of possible alternatives is significantly reduced.

3 XML and Regular Expressions Representation

XML is a metamarkup language that allows a document's structure to be described in terms of a hierarchy of named elements with additional properties expressed as attribute-value pairs. It provides platform for standard infrastructure in situations where text and other media are to be combined, exchanged and published [1]. Popularity of XML is strongly influenced by the World Wide

Web development: XML is designed to be used on the Web. This deployment on the Web opens up many new opportunities for using XML that was never available with SGML [4].

XML offers great flexibility in data structuring. It is often used as an import/export format or as a universal data exchange format. However XML can be used also directly within a data processing as an internal format. Complicated internal structures are often used to describe less or more complex entities and relations between them. We believe that XML can and should be exploited in many such cases instead internal structures. It will enhance interoperability of different systems (software agents in particular) [7]. Undoubtedly, XML alone does not provide the job itself. We need powerful tools to manipulate data represented by XML.

We can see the XML document as a tree of nodes with their defined attributes and values (many existing tools manipulate with the XML document in this way). Then various functions are designed for the manipulation and query such a tree and its nodes [6].

On the other hand, we can look and manipulate the XML document as a string (often long and complicated). Strings are effectively manipulated by the use of regular expressions. Regular expression is used to search or replace text fragments specified by a special syntax. The richness and power of regular expressions is acceptable for most of the XML processing tasks. Such processing can be effectively and easy expressed in several programming languages (e.g., Perl).

3.1 Perl: Comparison of Tree-Parsing Functions and Regular Expressions Usage

Perl is an efficient programming language for text processing. Perl has built-in regular expressions support. Together with lot of various libraries, portability and a free of charge policy, Perl is an effective tool for many text-processing tasks.

Perl is suitable for the use of combination of regular expressions and the XML. We used for this task the library `XML-Parser` aimed to parse and manipulate XML and the library `XML-DOM` that provides functions conforming to API of the Document Object Model. Various functions of these two libraries allow parsing of the XML document, selecting specified node (tag), modifying nodes' attributes, adding sub-nodes etc.

Simple example bellow shows the way of manipulation with the XML documents in Perl:

```
$xml = '<doc1><item1 attr1="val1">Hello</item1></doc1>';         #1
$root = $parser->parse($xml);                                    #2
$doc = $root->getElementsByTagName("doc1",0)->item(0);           #3
$item = $doc->getElementsByTagName("item1",0)->item(0);          #4
print $doc->getAttribute("attr1");                               #5
```

The first line defines the XML string. The parser parses this string and returns root node at the second line. After the successful parse we can manipulate the XML string, e.g. query child nodes by their name (lines 3 and 4) or work with the node's attribute (line 5).

We can also manipulate the XML document by the use of string functions for regular expressions. The following example matches the tag `<item1>` preceded by the tag `<doc1>` and remembers the value of the attribute `attr1`. Everything matched inside brackets `()` is automatically stored into special variables `$1`, `$2`, etc. Number after `$` sign is determined by the sequence of the opening bracket, so the contents of the first bracket pair is stored in the `$1` variable, contents of the second brackets will be in variable `$2` etc.

```
$xml =~ m{<doc1><item1[^>].*attr1="(.*?)"};
print $1;
```

It is clear that in some cases usage of regular expressions can be very efficient for writing, but it is limited to relatively simple manipulations.

4 Data and Knowledge for Syntactic Parser

We use XML and regular expressions to represent data and knowledge required for realisation of the method of syntactic analysis described in Section 2. Our approach to analysis of a sentence is based on subsequent refining of partial results. During the analysis different data and knowledge are used and subsequent result (tagged text) travels through defined stages. It would be advantageous to use a platform independent and extensible format, i.e. XML, for respective text analysis steps.

Figure 2 illustrates relation between data and particular steps of the syntactic analysis. Suffix analysis belongs to the step of morphological analysis. It is depicted also within sentence analysis frame, because our method of syntactic analysis actually does not need a base form of the word. We found out that result of combination of the suffix analysis and the syntactic analysis is satisfactory for several applications of the method.

4.1 Static View

It is obvious that we need a large repository of linguistic data and knowledge to perform parsing and checking a sentence. There are many ways how different authors solve the storage of these data and knowledge. They often use some self-defined text formats or database tables (see e.g. how word can be tagged [11]).

We use the XML language to represent all kinds of data and knowledge in the system. It is well-readable and easy manipulating format for both people and computers. The following example shows the record for the word in a dictionary, which represents pronoun *ja* (*I* in English):

```
<PRONOUN case="1" number="sg" gender="m|f|n" person="1">ja</PRONOUN>
```

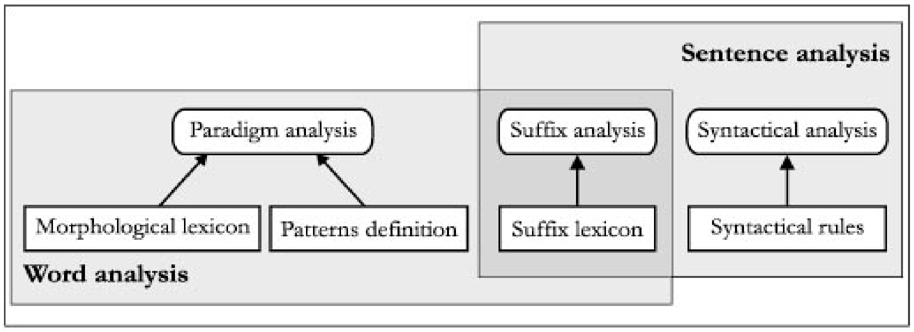

Fig. 2. Steps of the analysis and required lexicons.

Similarly we define rules for suffix analysis. The rule defines probability of a word form for its particular ending. The example bellow shows such rules for suffixes -*aca* and -*iaca*):

```
<NOUN cf="2" animate="y|n" case="2|4" number="sg" gender="m">+aca</NOUN>
<ADJECTIVE cf="2" case="1" number="sg" gender="f">+iaca</ADJECTIVE>
```

Although XML provides an efficient representation for lexicon definitions there is also another possibility how to use it within our system. XML can be used as an internal representation of the current state of the computation process. We use XML directly during parsing to manipulate already parsed fragments and replace them with a new one, also written in XML.

The figure 2 illustrates primary data and knowledge necessary to the analysis of a sentence. In order to achieve good results of the analysis, different statistics and heuristics should be represented.

For example, to perform an adaptive morphological analysis (mentioned in Section 2) we represent the statistical information about particular word forms in the following form: `<WORD occ="2" ref="1" sk="1">adresou</WORD>` (`occ` attribute represents total number of the particular word occurences).

However, for the sake of the efficiency lexicon data should be stored in a database. Data are extracted from the database (on the fly while being read from the database) and transformed into the XML document between the successive steps of the analysis.

4.2 Dynamic View

Regular expressions represent transformations of static structures, their modification, matching and manipulation. Matching expressions are used to categorise or check static elements. In a definition of incorrect words such matching rule can express non-existent letter combination.

For example, `<INCORRECT>m{b[bdfgpqwxz]}</INCORRECT>` means that word where letter `b` is followed by one of letters `b` or `d` or `f`, etc. is incorrect in the Slovak language .

More often we use substitution expressions that specify rewriting of the XML string during the analysis. Substitution expressions are used to express syntactic rules of the analysed language. Substitution expressions are stored inside a tag `RULE`. The rule consists of the two parts:

```
<RULE> s{left-hand-side}{right-hand side} </RULE>
```

Left-hand side of the rule determines text to be substituted. Right-hand side specifies the text used for a substitution. The following example rule specifies the combination of attribute and noun phrase with the same grammatical categories into one phrase:

```
<RULE cf="8.5">
 s{ <ATTRIBUTE (animate="\w*") (case="\d*") (number="\w*")
        (gender="\w*")>(\r*)  </ATTRIBUTE>
    <NOUNPHRASE \1 \2 \3 \4 (person="\d*")
        (compoundnumber="\w*")>(\r*)</NOUNPHRASE>  }
  { <NOUNPHRASE $1 $2 $3 $4 $6 $7>$5 $8</NOUNPHRASE>    }m
</RULE>
```

Rules are exploited on all levels of the analysis. On the morphological level, rules are used for paradigm specification. In this case the rule specifies transformations between various forms of a word. On the syntactical level, rules describe transformation of morphological categories of words and into phrases and phrases into a sentence.

4.3 Co-operation of the Static and Dynamic views

In order to provide useful results of syntactic analysis, the system should stick static and dynamic point of view together. Static structures represent necessary framework for information needed for parsing and also define status of the parsing process and different alternatives. On the other hand, regular expressions are used for representation of a dynamic behaviour of the system, i.e. how status structures are changing. The appendix demonstrates example of parsing phrase *"malá cesta"* (*"small road"* in English).

5 Conclusions

In the paper we describe our approach to syntactical analysis realisation by the use of XML and regular expressions. The main advantage of described approach is platform independence and extensibility. XML is used as a means for specifying data and knowledge needed for performing particular steps of the sentence analysis.

Our approach provides space for defining specific vocabulary for natural language analysis similarly to existing standards aimed to different application domains (e.g., MathML, a markup language for mathematical equations and formulae, GedML for marking up genealogical data, RDF (Resource Description

Format) describing a standard for representing information about web resources, etc.). We started this work by defining DTD's for data and knowledge needed for morphological and syntactical analysis.

We developed software tools to validate described approach. Two programs, *CheckText* and *VisualCheck* perform grammar checking of the Slovak language. The first one receives text for checking and returns the XML document, which represents a parsing tree of correct sentences and incorrect words or sentences. *VisualCheck* is the simple GUI program for interactive usage. It allows a user to write down some sentences and invoke the grammar checker. Then program highlights errors and suggests corrections to the user.

Important part of our system is an agent-based subsystem used for automatic intelligent expansion of system's morphological lexicon. It is based on a central synchronisation server and group of independent agents that browse Internet web pages and collect various forms of Slovak words. They exchange collected information through the central server. Such agents can be a part of grammar checker on the user's machine and can perform self-improvement of the sentence analysis as the background process [10].

One of the most interesting areas of usage of natural language analysis is an intelligent database interface. Such interfaces allow users to query database using commands and queries written in natural language and bring the power of querying to the wider range of users. In fact, natural language interface should not be limited to databases only, it is useful in almost any information system.

Also searching in texts written in other languages than English often requires additional pre-processing based on the linguistic knowledge. Various word forms in inflectional languages make it hard to find in texts all forms of a word. As a result a user often receives only a small portion of appropriate results. Another promising application for natural language analysis is information filtering. Our realisation based on XML enables to incorporate content based filtering into the existing collaborative filtering approaches [9].

Proper use of linguistic knowledge and tools in information systems can provide easier user interface and interesting new functionality. The use of XML with regular expressions brings many advantages including simplicity, platform and application independence and extensibility.

References

1. Bradley. N. *The XML companion.* Addison Wesley, Harlow, England, 1998.
2. Clark, D. Natural language relevancy ranking, and common sense. *IEEE Intelligent Systems*, July/August 1999, 17-19.
3. Cowie, J., Lehnert, W. Information Extraction. *Communications of the ACM*, Vol. 39, No. 1, 80-91, January 1996.
4. Garshol, L.M. What can we do with XML? White paper, Infotek A/S, paper presented at the SGML/XML Finland '98 conference in Jyväskylä and at SGML User Group meeting in Oslo, 1999. Available also at $http : //www.stud.ifi.uio.no/^{\sim}lmariusg/download/artikler/fin_sgml_98.html.$

5. Hausser, R. *Newcat: Parsing Natural Language using Left-Associative Grammar.* Berlin, Springer-Verlag, 1986.
6. Laforest, F., Tchounikine, A. A Model for Querying Annotated Documents. In Proc. of Advances in Databases and Information Systems, ADBIS'99, J. Eder et. al. (Eds.), Springer Verlag, LNCS 1691, pp. 61-74, 1999.
7. Leskovar, R.T., Györkös, J. Interoperability in an Agent-based Workflow System. In Proc. of Short Papers Advances in Databases and Information Systems, ADBIS'99, J. Eder, I. Rozman, T. Welzer (Eds.), pp. 88-94, 1999.
8. Smrž, P., Horák, A. Implementation of Efficient and Portable Parser for Czech. In Proc. of the Second Int. Workshop, TSD'99, V. Matoušek et. al. (Eds.), Springer Verlag, LNAI 1692, pp. 105-108, 1999.
9. Polčicová, G. Recommending HTML-documents using Feature Guided Automated Collaborative Filtering. In Proceedings of Short Papers Advances in Databases and Information Systems, ADBIS'99, J. Eder et. al. (Eds.), pp. 81-87, 1999.
10. Trabalka, M., Bieliková, M. Performing Adaptive Morphological Analysis Using Internet Resources. In Proc. of Text, Speech and Dialog, TSD'99, V. Matoušek et. al. (Eds.), Springer Verlag, LNAI 1692, pp. 66-71, 1999.
11. Nenadic, G., Vitas, D. Using Local Grammars for Agreement Modeling in Highly Inflective Languages. In Proc. of Text, Speech and Dialog, TSD'98, P. Sojka et. al. (Eds.), Masaryk University Press, pp.91-96, 1998.
12. Páleš, E. Sapfo. *Paraphraser of Slovak Language.* Bratislava, 1993. (In Slovak)

Appendix: Example of Parsing

Input: `<TEXT>malá cesta</TEXT>`

Output of morphological analysis:

```
<ADJECTIVE animate="" case="1" number="sg" gender="f">malá</ADJECTIVE>
<NOUN animate="" case="1" number="sg" gender="f">cesta</NOUN>
```

Output of 1st syntactic analysis step:

```
<ATTRIBUTE animate="" case="1" number="sg" gender="f">malá</ATTRIBUTE>
Used rule:
<RULE cf="10">
   s{<ADJECTIVE (animate="\w*") (case="\d*") (number="\w*")
       (gender="\w*")>(\r*)</ADJECTIVE>    }
     {<ATTRIBUTE $1 $2 $3 $4>$5</ATTRIBUTE>      }m
</RULE>
```

Output of 2nd syntactic analysis step:

```
<NOUNPHRASE animate="" case="1" number="sg" gender="f" person="3">
malá cesta</NOUNPHRASE>
Used rule:
<RULE cf="8.5">
     s{<ATTRIBUTE (animate="\w*") (case="\d*") (number="\w*")
        (gender="\w*")>(\r*)</ATTRIBUTE>
               <NOUN \1 \2 \3 \4>(\r*)</NOUN>   }
          {<NOUNPHRASE $1 $2 $3 $4  person="3">$5 $6</NOUNPHRASE>  }m
</RULE>
```