A Hybrid Strategy for Regular Grammar Parsing

Kiril Simov and Petya Osenova

BulTreeBank Project
http://www.BulTreeBank.org
Linguistic Modelling Laboratory, Bulgarian Academy of Sciences
Acad. G. Bonchev St. 25A, 1113 Sofia, Bulgaria
kivs@bultreebank.org, petya@bultreebank.org

Abstract

The paper outlines a hybrid architecture for a partial parser based on regular grammars over XML documents. The parser is used to support the annotation process in the BulTreeBank project. Thus the parser annotates only the 'sure' cases. To maximize the number of the analyzed phrases the parser applies a set of grammars in a dynamic fashion. Each grammar determines not only the constituent structure (plus some syntactic dependencies internal to the structure), but also a description of the local and global context of the recognized phrase. The grammars available to the parser are arranged in a network. The order of the grammars application depends on the initial ordering in the network and the descriptions associated with the grammars. Thus the traverse is not deterministic. Additionally, the application of the grammars can be interleaved with the applications of other XML tools like *remove*, *insert* and *transform* operations. This architecture provides a flexible means for guiding the linguistic analysis in order to utilize all the available linguistic knowledge and to produce a very accurate partial analysis.

1. Introduction

The creation of a treebank is an enormous effort requiring a lot of manual work during the annotation and validation of the linguistic data. In order to minimize this kind of work the annotation architecture of BulTreeBank project relies on a large set of preprocessing tools like *morphological analyzer*, *partial grammars*, *named entities module*, etc. In order to be reliable with respect to the next processing stages and the validation procedure, we require each of the preprocessing steps to recognize only correct pieces of information. In this paper we present our partial parser. It is based on regular grammars over XML documents.

The XML representation of the analyses allows us to check whether certain conditions hold before the application of a given grammar over the content of some XML element. These conditions are represented as a description over the XML tree containing the element which the grammar will be applied to. This provides us with a means for a detailed selection of which grammar to be applied where and in which order. Also, XML allows different ways of incorporating the grammar application results. Another advantage of the XML technology is that the application of the grammars does not require a strict bottom-up or top-down strategy, it permits dynamic changes at the level of the grammar application selecting different nodes in the XML tree. Additionally, we can use other tools over XML documents in order to perform operations which are not accessible to the grammar engine, such as *removing*, *inserting* or *transforming* XML elements, attributes or fragments.

Here we present an architecture for a *dynamic reordering* of the regular grammars application for parsing. The architecture is implemented within the CLaRK System ((Simov et. al. 2001), http://www.bultreebank.org/clark/index.html) over an XML representation of the processed data.

The structure of the paper is as follows: next section provides a brief overview of some techniques for partial parsing with regular grammars; section 3. describes the regular grammars in CLaRK system; section 4. presents the dynamic architecture of regular grammars; next section focuses on the elements of the implemented parser. The last section concludes the paper.

2. Regular Grammars and Partial Parsing

In this paper we consider regular grammars as a basic device for achieving partial parsing. Although the regular grammars are not the only possible way to construct a partial parser, it remains the most well-known one. Regular grammars are appealing to the NLP community because of their effectiveness and applicability. It has been studied which phenomena can be successfully described by them and which strategies can be used for the implementation of the desired behaviour. In NLP there are two competing strategies at the pre-processing level: bottom-up (easy-first) and top-down (large-first).

The chunks, being bottom-up units, were defined first as 'the non-recursive core of an intra-clausal constituent to its head, but not including post-head dependents' (Abney 1996). Abney's strategy relies on three heuristics: detecting 'islands of certainty', applying 'easy-first parsing' and 'preferring syntax to semantics'. Hence, in Abney's version chunking suggests cascaded bottom-up strategy mainly. The critics of this approach are concerned with the insufficient coverage and misattachments.

At the opposite extreme is the pure top-down approach (van Delden and Gomez 2003), which relies on the help of the punctuation, conjunctions and other markers of delimitation. This approach claims that instead of delaying the attachment resolution, it handles it immediately. However, this approach is applicable only in the presence of certain markers. It identifies the general clausal structure within the sentence and thus restricts the proliferation of the possible syntactic analyses. However, top-down based approaches do not explain what happens when these markers are absent or misplaced in the text.

There have been efforts for combining the advantages

of both approaches. For example, adding some top-down filtering upon chunking level and establishing a connection with shallow parsing (Müller 2002). Another example is the Topological fields approach to German (Frank et. al. 2003) which first annotates the macrostructure of the sentences and localizes the syntactic relations within topological fields of the German sentence. To avoid the strict bottom-up (Abney 1996) or top-down (van Delden and Gomez 2003) strategy, their systems use temporary tagging which blocks the application of certain rules.

Another important mechanism in the construction of partial parsers is the 'modularity' of the grammars (cascaded grammars). In fact, the parser uses a sequence of grammars where each grammar in the sequence works over the result of the previous grammar work (if any). This mechanism allows the grammars used in the parser to be tuned to certain types of sentences. An example for such tuning is presented in (Nuria 2001) who proposes a 'two-tier' analysis with respect to the complexity of the sentences in the corpus. Then the grammars are divided into a 'core' part which is applicable to simpler sentences common to all corpora and 'specific' part of grammars dealing with more complex language phenomena. The sentences are divided into the two categories (*first-tier* (easier) and *second-tier* (harder) sentences) after the partial parsing depending on the recognized elements and the presence of punctuation, conjunctions, etc. After the classification the partial analyses for the second-tier sentences are removed and they are re-analyzed with a set of specific grammars.

The distinction 'bottom-up' vs. 'top-down' methods interleaves with the dichotomy 'constructive approach' vs. 'reductionist approach'. The former concentrates on identifying the basic phrases with the help of local grammars, while the latter starts from a set of alternatives and then reduces them via constraints. In (Ait-Mokhtar and Chanod 1997) some merging techniques over both approaches are introduced and the non-monotonicity of the parser is stressed.

In our opinion, more effort should be put on merging strategies and mechanisms, because the combination of different sets of advantages gives the best results. Summing up the above discussion and defining some additional requirements, we think that for maximizing the usage of the available linguistic knowledge we need a parser which allows for at least the following functionalities: (1) application of the grammar rules in a cascaded fashion; (2) context dependent application of the grammar rules; (3) usage of temporary annotation (non-monotonicity); (4) change of the parsing strategy: mixing top-down and bottom-up parsing; (5) dynamic reordering of the grammars application.

In the rest of the paper we describe the means that are used for the construction of a partial parser for Bulgarian with the above functionalities.

3. Regular Grammars in CLaRK System

The CLaRK System is an XML-based system for corpora development – see (Simov et. al. 2001). It incorporates the following technologies: *XML technology*; *Unicode*; *Regular (Cascaded) Grammars*; *Constraints over XML Documents*. For document management, storing and querying, we chose the XML technology because of its popularity and its ease of understanding. The core of CLaRK is an Unicode XML Editor, which is the main interface to the system. Besides the XML language itself, the system implements an XPath language engine for navigation in documents and an XSLT engine for transformation of XML documents. The XSL transformations can be applied locally to an XML element and its content. There is a mechanism for the creation of a hierarchy of tokenisers. They can be attached to the elements in the DTDs and in this way there are different tokenizers for different parts of the documents. Constraints can be used in two modes: *insertion* and *validation*. Other tools are: remove operation, XPath insertion, sorting, extraction, concordance, etc.

The regular grammars in CLaRK System work over token and element values generated from the content of an XML document and they incorporate their results back in the document as XML mark-up (Simov, Kouylekov and Simov, 2002). The tokens are determined by the corresponding tokenizer. The element values are defined with the help of XPath expressions, which determine the important information for each element. In the grammars, the token and element values are described by token and element descriptions. These descriptions could contain wildcard symbols and variables. The variables are shared among the token descriptions within a regular expression and can be used for the treatment of phenomena like syntactic agreement. The grammars are applied in a cascaded manner. The general idea underlying the cascaded application is that there is a set of regular grammars. The grammars in the set are in a particular order. The input of a given grammar in the set is either the input string, if the grammar is first in the order, or the output string of the previous grammar. The evaluation of the regular expressions that define the rules, can be guided by the user. We allow the following strategies for evaluation: 'longest match', 'shortest match' and several backtracking strategies.

Here is an example, which demonstrates the cascaded application of two grammars. The first grammar consists of the following rule:

```
<np aa="NPns">\w</np> ->
     <("An#"|"Pd@@@sn")>,
     <("Pneo-sn"|"Pfeo-sn")>
```

Here the token description[1] `"An#"` matches all morphosyntactic tags for adjectives of neuter gender, the token description `"Pd@@@sn"` matches all morphosyntactic tags for demonstrative pronouns of neuter gender, singular, the description `"Pneo-sn"` is a morphosyntactic tag for the negative pronoun, neuter gender, singular, and the description `"Pfeo-sn"` is a morphosyntactic tag for the indefinite pronoun, neuter gender, singular. The brackets < and > delimit the element descriptions within the rule. This rule recognizes as a noun phrase each sequence of two elements where the first element has an element value corresponding to an adjective or demonstrative pronoun with appropriate grammatical features, followed by an element with element value corresponding to a negative or an indefinite pronoun.

[1] Here # and @ are wildcard symbols.

Notice the attribute aa of the rule's category. It represents the information that the resulting noun phrase is singular, neuter gender. Let us now suppose that the next grammar aims at the determination of prepositional phrases and it is defined as follows:

```
<pp>\w</pp> -> <"R"><"N#">
```

where "R" is the morphosyntactic tag for prepositions. Let us trace the application of the two grammars one after another on the following XML element:

```
<text>
  <w aa="R">s</w>
  <w aa="Ansd">golyamoto</w>
  <w aa="Pneo-sn">nisto</w>
</text>
```

First, we define the element value for the elements with tag w with the XPath expression: "**attribute::aa**". Then the cascaded regular grammar processor calculates the input word for the first grammar: "<" "R" ">" "<" "Ansd" ">" "<" "Pneo-sn" ">". Then the first grammar is applied on this input words and it recognizes the last two elements as a noun phrase. This results in two actions: first, the markup of the rule is incorporated into the original XML document:

```
<text>
  <w aa="R">s</w>
  <np aa="NPns">
      <w aa="Ansd">golyamoto</w>
      <w aa="Pneo-sn">nisto</w>
  </np>
</text>
```

Second, the element value for the new element <np> is calculated and it is substituted in the input word of the first grammar. In this way the input word for the second grammar is constructed: "<" "R" ">" "<" "NPns" ">". Then the second grammar is applied on this word and the result is incorporated in the XML document:

```
<text>
  <pp>
     <w aa="R">s</w>
     <np aa="NPns">
         <w aa="Ansd">golyamoto</w>
         <w aa="Pneo-sn">nisto</w>
     </np>
  </pp>
</text>
```

The following rule demonstrates the usage of variables in a rule:

```
<np aa="NP&G&N">\w</np> ->
  (<"A&G&Nd">,<"A&G&Ni">*)?,<"N@&G&Ni">
```

Here &G and &N are variables whose use will ensure the agreement in gender and number. The variables can take as values arbitrary non-empty strings within a token. Additionally, the user can define a domain for a certain variable (a set of permissible values) and a negative domain (a set of values which are not allowable). In the example above, the domain for variable &G can be: f, m or n (standing for feminine, masculine and neuter gender). If no (positive) domain is defined then the variable can have any string, which is not presented in the negative domain, as a value. The rule itself says that an np is a sequence of a definite adjective followed by any number of indefinite adjectives and an indefinite noun. The variable ensures the agreement in gender and number and their values are copied to the resulting annotation for the np.

The target of a grammar application is determined by an XPath expression, we call this expression a *target description*. The grammar is applied over the context of the elements described by the target description (selected by the XPath expression). This possibility gives us a flexible way to determine where to apply the grammar depending on the context of the elements. Another mechanism offered by the system is the filtering of the input for the grammar. We are able to hide some elements from the content of the element which the grammar is applied to. In this way, for example, we can hide some parenthetical expressions when they are inside some chunk.

4. Towards a Dynamic Network of Grammars

Facing the linguistic data, it turns out that the method of cascadedness is not sufficient, because of two reasons: 1. sometimes the analyses are performed in a non-monotonic way whereas the cascaded method allows only strict ordering from one level to another and 2. sometimes the application of one grammar influences the applicability of the others. Thus we need a network of grammars with cycles in order to find the best way to process the data.

Our idea is close to the one presented in (Nuria 2001) for merging determinism with incrementality. A syntactic diagnosis is needed before the application of an adequate modular approach. The difference lays in the following: we do not divide the parsing modules into treating simple and complex structures. We rather extend this assumption with dynamic switches between different structures being either simple, or complex.

The mechanisms in CLaRK allow us to construct a dynamic network of grammars: a set of very specific versions of nearly the same grammar which is applied in different contexts. They mainly differ in the results that are incorporated back in the XML document, in the way the system prepares the input for them, their target descriptions and their filters. For example, the nominal chunker (Osenova 2002) is divided into two modules according to the context: 'contextually non-dependent' and 'after preposition'. In this way the noun groups with uncertain starting indicators are analyzed only in 100 % sure contexts which is their position after prepositions. As a result, coverage suffers, but a high accuracy is achieved.

We can determine the order of the grammar applications in two ways: *fixed order networks* — a set of ordered grammars in which a grammar is applied after the preceding grammar has been applied; *concurrent order networks* — a set of grammars or grammar networks (with fixed or concurrent order) where each grammar or a grammar network

is associated with an XPath expression. If the XPath expression is satisfied over the XML document then the corresponding grammar or a network can be applied. The XPath expressions are also ordered. The system checks the XPath expressions for satisfiability in turn and for the first satisfied expression it applies the corresponding grammar (or network). After the application of a grammar (or a network) the system can either continue with next expressions in the sequence, to stop the processing, or to change the point in the sequence from where to proceed further. The specificity of the context description is defined before the application. Hence the clashes between the context descriptions of the grammars is a responsibility of the grammar writer. The order imposed over the XPath expressions is used for avoiding clashes. This procedure allows us to implement any strategy of evaluation over the different grammars. The target descriptions of the grammars determine the bottom-up or the top-down application. The context descriptions determine a dynamic order of applications depending on the linguistic features presented in the analysis so far and the results of the previous processing.

The application of the grammar can interleave with the application of other tools of the system like *remove*, *insert* and *transform* ones. In this way one can introduce temporary annotation in the document which to be deleted later. Also, when checking for certain configurations in the document, some transformation can be applied locally. In this sense our way of processing is non-monotonic.

5. Processing the Data

The actual processing in the annotation of the Bulgarian sentences is organized in three stages:

Easy-first treatment (bottom-up). The non-recursive easy-first parsing is applied to base noun phrases, adjectival phrases, adverbial phrases and verbal complexes. Here we include lexical patterns with heads plus clitic (definite noun plus possessive or interrogative clitic, definite adjective plus possessive or interrogative clitic etc.). Note that the verb can be combined with several clitics in sequence. Also, some of the grammars for named entities, idiomatic expressions, dates, abbreviations, multiwords are applied during this step of processing.

Large-first treatment (top-down). The large-first technique is applied in two ways: (1) identifying larger groups of elements and (2) by a list of fixed expressions. The former is applied preferably to some clauses, which have clear starting markers and supporting punctuation like: relative clauses, wh-clauses, clauses for purpose and cause. The latter maps the fixed expressions to the tokens by means of grammars for parentheticals, introductory phrases. Some fixed expressions are considered markers for clausal detection. For example, the verbs of saying take whole sentences as complements in direct speech contexts. These clauses are easily identifiable by the presence of the appropriate punctuation.

Network-based treatment. The processing of some linguistic segments cannot be performed properly neither with bottom-up, nor with top-down cascaded strategies only. It needs the appropriate combination of both methods. Typical examples are PPs, Da-clauses (Bulgarian infinite clauses), coordinations, different clausal embeddings, all clausal boundaries. PPs can be identified in a bottom-up manner in sure positions, but with different contexts: (1) when sentence-final, and (2) within already identified clauses. Da-clauses are identified precisely when trapped within other clauses. Thus, when a grammar for other kinds of clauses succeeds, the grammar for da-clauses is applied. A similar strategy is applied to the relative clauses when their attachment depends on the annotation of other clauses. Coordination is processed better within already identified clausal boundaries. The grammars can also handle some discontinuity phenomena like extraction. For example, there are patterns, in which the subject of the da-clause is extracted in front of the heading impersonal verb. The subjecthood of the extracted noun phrase can be easily checked by mapping person and number features.

6. Conclusion

In this paper we presented an architecture for a (non-monotonic) partial parsing. The main characteristics of the architecture is the dynamism of the grammars application depending on the context. The utility of this hybrid strategy is proved during the annotation of the sentences in the BulTreeBank project.

7. References

Abney. 1996. *Chunk Stylebook.* On http://sfs.nphil.uni-tuebingen.de/ abney/Papers.html, draft.

Ait-Mokhtar S. and Chanod J-P. 1997. *Incremental Finite-State Parsing.* In: *Proc. of the 5th Conference of Applied Natural Language Processing. ACL*. USA. pp. 72–79.

Frank A., Becker M., Crysmann B., Kiefer B., Shäfer U. 2003. *Integrated Shallow and Deep Parsing: TopP meets HPSG*. In: *Proc. of 41st ACL Conference.* Japan. pp. 104–111.

Müller. 2002. *Shallow-Parsing Stylebook for German.* On http://www.sfs.nphil.uni-tuebingen.de/dereko/anno-doc.html

Nuria Gala Pavia. 2001. *A two-tier corpus-based approach to robust syntactic annotation of unrestricted corpora.* In: *Traitement Automatique des Langues*, Special Issue on Corpus Linguistics. Vol. 42, No 2.

Petya Osenova. 2002. *Bulgarian Nominal Chunks and Mapping Strategies for Deeper Syntactic Analyses.* In: *Proc. of The Workshop on Treebanks and Linguistic Theories*. Sozopol, Bulgaria.

Kiril Simov, Zdravko Peev, Milen Kouylekov, Alexander Simov, Marin Dimitrov, Atanas Kiryakov. 2001. *CLaRK - an XML-based System for Corpora Development.* In: *Proc. of the Corpus Linguistics 2001 Conference.* pp 558–560.

Kiril Simov, Milen Kouylekov, Alexander Simov. 2002. *Cascaded Regular Grammars over XML Documents.* In: *Proc. of the 2nd Workshop on NLP and XML (NLPXML-2002).* Taiwan.

Sebastian van Delden and Fernando Gomez. 2003. *A Larger-first Approach to Partial Parsing.* In: *Proc. of RANLP'03 Conference.* Bulgaria. pp 124–131.