Development of Corpora within the CLaRK System
The BulTreeBank Project Experience

Kiril Simov, Alexander Simov, Milen Kouylekov,
Krasimira Ivanova, Ilko Grigorov, Hristo Ganev
BulTreeBank Project
Linguistic Modelling Laboratory - CLPPI, BAS, Sofia, Bulgaria
kivs@bultreebank.org, adis_78@dir.bg, mkouylekov@dir.bg,
krassy_v@abv.bg, ilko_grigorov@yahoo.com, hristo_ganev79@yahoo.com

Abstract

CLaRK is an XML-based software system for corpora development. It incorporates several technologies: XML technology; Unicode; Regular Cascaded Grammars; Constraints over XML Documents. The basic components of the system are: a tagger, a concordancer, an extractor, a grammar processor, a constraint engine.

1 Introduction

The CLaRK System is an XML-based system for corpora development – see (Simov et. al., 2001). The main aim behind the design of the system is the minimization of human intervention during the creation of language resources. 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, we implemented an XPath language 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.

For multilingual processing tasks, CLaRK is based on an Unicode encoding of the text inside the system. 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 tokenisers for different parts of the documents.

The basic mechanism of CLaRK for linguistic processing of text corpora is the cascaded regular grammar processor. The main challenge to the grammars in question is how to apply them on XML encoding of the linguistic information. The system offers a solution using the XPath language for constructing the input word to the grammar and an XML encoding of the categories of the recognised words.

Several mechanisms for imposing constraints over XML documents are available. The constraints cannot be stated within the standard XML technology. The constraints are used in two modes: checking the validity of a document regarding a set of constraints; supporting the linguist in his/her work during the building of a corpus. The first mode allows the creation of constraints for the validation of a corpus according to given requirements. The second mode helps the underlying strategy for minimisation of the human labour.

We envisage several uses for our system: *Corpora markup.* Here users work with the XML tools of the system in order to mark-up texts with respect to an XML DTD. This task usually requires an enormous human effort and handles both - the mark-up itself and its validation afterwards. Using the available grammar resources, such as morphological analyzers or partial parsing, the system can state local constraints reflecting the characteristics of a particular kind of texts

or mark-up. *Dictionary compilation for human users.* The system supports the creation of actual lexical entries, whose structure can be defined via an appropriate DTD. The XML tools can be used also for corpus investigation that provides appropriate examples of the word usage in the available corpora. The constraints, incorporated in the system, can be used for writing a grammar over the sublanguages of the definitions, for imposing constraints over elements of lexical entries and the dictionary as a whole. *Corpora investigation.* The CLaRK System offers a rich set of tools for searching over tokens and mark-up in XML corpora, including cascaded grammars, XPath language. Their combinations are used for tasks, such as: extraction of elements from a corpus - for example, extraction of all NPs in the corpus; concordance - for example, viewing all NPs in the context of their use.

The first version of the CLaRK System was released on 20.05.2002 and it is freely available at the site of the BulTreeBank Project[1]. It is actively used within the BulTreeBank Project for maintenance of language resources of different kinds – text archive, morphologically annotated corpora, syntactic trees and lexicons. It is implemented in Java and was tested under MS Windows and Linux.

The paper describes three applications of the CLaRK system, related to corpora development. These includes: chunk grammars, disambiguation and evaluation (precision and recall). This paper does not discuss related work due to space limitations.

2 Cascaded Regular Grammars

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. 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 agreement. The grammars are applied in 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 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 defines 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 (based on a grammar developed by Petya Osenova for Bulgarian noun phrases):

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

Here the token description[2] `"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 > delimits 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. 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 is for determination of prepositional phrases and it is defined as follows:

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

[1] http://www.BulTreeBank.org/clark/index.html

[2] Here # and @ are wildcard symbols.

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 by 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 and 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>
```

Because the cascaded grammar consists of only these two grammars, the input word for the second grammar is not modified, but simply deleted.

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 for one symbol only and ensure the agreement on gender and number. Additionally, there are variables for strings (denoted as &&N). For each variable a set of constraints can be imposed via regular expressions[3]. Note that the variables are used also within the XML mark-up and their values will be incorporated within the document when this rule recognizes some word in the document.

3 Using Constraints for Manual and Automatic Disambiguation

Here we demonstrate the constraints of type "Some Children". This kind of constraints deals with the content of some elements. They determine the existence of certain values within the content of these elements. A value can be a token or an XML mark-up and the actual value for an element can be determined by the context. Such a constraint works in the following way: first it determines to which elements in the document it is applicable (the conditions over the context of the nodes are expressed by an XPath expression), then for each such element in turn it determines which values (usually they also are pointed by an XPath expression) are allowed and checks whether in the content of the element some of these values are presented as a token or an XML mark-up. If there is such a value, then the constraint chooses the next element. If there is no such a value, then the constraint offers to the user a possibility to choose one of the allowed values for the element and the selected value is added to the content.

Within BulTreeBank, we use these constraints for manual disambiguation of morpho-syntactic tags of wordforms in the text. For each wordform we encode the appropriate morpho-syntactic information from the dictionary as two elements: <aa> element, which contains a list of morpho-syntactic tags for the wordform separated by a semicolon, and <ta> element, which contains the actual morpho-syntactic tag for this use of the wordform. The value of <ta> element has to be among the values in the list presented in the element <aa> for the same wordform. "Some Children" constraints are very appropriate in this case. Using different conditions and filters on the values, we implemented and used more than 70

[3] In future work we envisage more complicated constraints to be implemented.

constraints during the manual disambiguation of wordforms in the "golden standard" of the project. It is important to be mentioned that when the context determines only one possible value, it is added automatically to the content of <ta> element and thus the constraint becomes a rule.

Another feature of the constraints is the usage of variables with the XPath expressions and the possibility for work with more than one document. This allows the lexicons used for annotation to be saved in a separate XML document and to be accessed when it is necessary.

4 Evaluation: Precision and Recall

At the moment the CLaRK system does not have a separate tool for comparing two XML documents, which to be used for calculation of the two measures – precision and recall. However, one could simulate it by using the present tools of the system. Let us have developed an NP chunk grammar and additionally, a manually annotated corpus. Let the NPs in the corpus have an attribute `type` with value `"m"`. In addition, we run the NP chunk grammar over a clean version of the corpus and the NPs in the result have the attribute `type` with value `"g"`. In order to evaluate the precision and recall of the grammar over the annotated corpus, we have to do the following:

1. First, we extract the NPs, which are presented within the corpus and in the output from the application of the grammar. Then we join the two sets of NPs in one XML document. With a perfect grammar, for each NP marked with value `"m"` there should be one NP marked with value `"g"` and vice versa, but usually this is not the case.

2. In order to find the discrepancies, we remove the NPs in pairs on the basis of the equal content. The only difference is that one NP in the pair has value `"m"` and the other one has value `"g"` for the attribute `type`. In the end, within the document, the NPs with attribute `type="g"` are those NPs that are recognized by the grammar, but there is no corresponding NPs in the corpus and similarly, the NPs with attribute `type="m"` are those NPs that are in the corpus, but they are not recognized by the grammar.

3. Then we count the two kinds of NPs, which have left unmatched in the document. We use the figures (together with the number of all NPs in the corpus) in order to calculate the precision and recall for the grammar.

Although this procedure is precise with respect to the internal structure of the compared sub-trees, it is not sensitive to the context of appearance of these sub-trees[4]. For example, one NP in the corpus can match a NP in the grammar result even if they are in quite different contexts. In order to solve the problem, we add to each leaf element in the XML documents unique identifiers that are the same for the corpus and the result from the application of the grammar. In this way we compare the NPs (in our case) on the basis of their content and also their position in the linear order of the words in the sentences.

5 Conclusion

The demonstration of CLaRK will show the basic tools of the system in the process of creating a linguistically interpreted text corpus of Bulgarian. Future directions of development are: implementing more of XML technologies (XML Schema, XPointer, XLink), implementation of a macro language, database support.

Acknowledgements

The work reported here is done within the BulTreeBank project. The project is funded by the Volkswagen Stiftung, Federal Republic of Germany under the Programme "Cooperation with Natural and Engineering Scientists in Central and Eastern Europe" contract I/76 887.

References

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, pages: 558-560.

[4] We would like to thank Csaba Oravecz and Tamás Váradi for pointing us to this problem.