The CLaRK System: XML-based Corpora Development System for Rapid Prototyping

Kiril Simov, Alexander Simov, Hristo Ganev, Krasimira Ivanova, Ilko Grigorov

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

Abstract

The paper presents the CLaRK System as a tool for the creation of XML-based corpora and a platform for rapid prototyping. The system provides a set of basic tools for processing XML documents. These tools include: tokenizers, regular grammars, constraints; remove, insert, extract, sort, transformation operations. Additionally, the system is equipped with a macro language which allows the creation of tools sequences. The macro language includes a set of control operators for guiding the application of the tools in the macro. Usually, a tool or a macro works over a single document changing it or producing a new document. In some cases processing of more than one document is necessary — in iterative statistics for treebank transformation, stand-off annotation, etc. For such processing the macro language allows a dynamic change of the processed documents.

1. Introduction

Many NLP tasks require the creation of manually annotated corpora. The same is true for more standardized corpora of less-processed languages. In this case, any support for (semi)automatic solution of some subtasks during the creation of the corpus is valuable. For some of the standard tasks there are ready (off-the-shelf) solutions that can be used, but for considerable problems there is not a standard solution because the problem is specific for the corresponding corpus. On the other hand, the corpus creators can find a solution during the annotation (for example, a new rule for disambiguation in a certain context). Many solutions of this kind are never implemented due to several reasons: the annotation system does not allow introduction of new kinds of rules or the implementation of new software modules is unfeasible. The CLaRK System presented here was designed to provide the user with a possibility for the creation of new processing tools on-the-fly, in the process of annotation.

The CLaRK System is an XML-based system for developing and exploration of text corpora. One of the main purposes which stands behind the design of the system is reducing the human labor during the creation of language resources. The system offers different facilities for encoding some regularities and dependencies in order different processing procedures to be run semi or full automatically.

For its work the system relies on the following key technologies: XML technology; Unicode; Regular Cascaded Grammars; Constraints over XML Documents. The architecture of CLaRK system is based on a Unicode XML Editor, which is the main interface to the system. Besides the XML language itself, the system supports an XPath language processor for navigation in documents and an XSLT engine for transformation of XML documents. Additionally to the standard way of XSL transformations application, there is a mechanism for applying locally transformations to XML elements, and their content and incorporating the results back in the source document.

CLaRK is based on a Unicode encoding of the text inside the system. For the purposes of segmenting the text in a sensible way there is a mechanism for creating of a hierarchy of tokenizers. They can be attached to the elements in the DTDs and in this way different tokenizers can be responsible 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 recognized word categories.

Several mechanisms for imposing constraints over XML documents are available. They 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 minimization of the human labor.

These basic tools can be combined in procedures via a macro language. Thus a sequence of processing steps can be done without the intervention of the user. The macro language is structured at two levels. At the first level all processing steps are carried out on a single document. At the second level different processing steps could work on different documents. At each of these levels the macro language provides `IF-THEN` and `GOTO` operators which ensure conditional application of the corresponding processing. Equipped with these macro language the system provides an environment for construction of processing module on-the-fly. The user can write appropriate annotation tools to facilitate the work or just to construct a prototype

to prove some new idea, before going to invest much time in its implementation.

In this paper we first describe the basic tools, then the macro language and then provide examples of application of the system features, outlined above.

2. Basic Tools of the System

In this section we present some of the basic tools of the CLaRK System for processing corpora. They include tokenization, (cascaded) regular grammars, constraints, transformation, remove, insert operations.

2.1. Tokenizers

The representation of the XML documents inside the system is based on UNICODE. This allows the representation of texts in different languages in a natural way. In XML each text node is considered a sequence of letters (characters). Unfortunately, that is unacceptable for corpus processing where one usually requires to distinguish wordforms, punctuation and other tokens in the text. In order to handle this problem the CLaRK System supports a user-defined hierarchy of tokenizers. At the very basic level the user can define a tokenizer in terms of a set of token types. In this basic tokenizer (known as *Primitive* tokenizer) each token type is defined by a set of UNICODE symbols. Above this basic level tokenizers the user can define other tokenizers for which the token types are defined as regular expressions over the tokens of some other tokenizer, the so called parent tokenizer. Tokens in the system are used in different processing modules. For each tokenizer an alphabetical order over the token types is defined. At the level of primitive tokenizers the system supports a means for token normalization. This is important for the cases when no distinction has to be made between tokens of different type. Such a case can be the comparison of tokens (words) containing letters in upper and lower case, where usually the case should be ignored.

Sometimes in different parts of one document the user might want to apply different tokenizers. For instance, in a multilingual corpus the sentences in different languages will need to be tokenized by different tokenizers. In order to allow this functionality, the system allows for attaching tokenizers to the documents via the DTD of the document. To each DTD the user can attach a tokenizer which will be used for the tokenization of all textual elements in the documents corresponding to the DTD. Additionally, the user can overwrite the DTD tokenizer for some of the elements attaching to them other tokenizers.

2.2. 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 an XML mark-up (see (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 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 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 defines the rules, can be guided by the user. We allow the following strategies for evaluation: 'longest match', 'shortest match' and several backtracking strategies. 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). If no (positive) domain is defined then the variable can have any string, which is not presented in the negative domain, as a value.

2.3. Constraints over XML Documents

In this section we present the three kinds of constraints which can be imposed over XML documents in the CLaRK System (see (Simov, Simov and Kouylekov 2003)).

2.3.1. Value Constraints

There are four types of these constraints: *parent*, *all children*, *some children*, *some attributes*. In *validation mode* all of them check whether the closest surrounding of the node satisfies some conditions. In *information entering support mode* they offer to the user a possibility for entering information in order to satisfy the constraint.

Parent constraints put additional restrictions over the possible parent of a node. In the validation mode they are checked together with the constraints that are imposed in the DTD. Otherwise, parent constraints apply when the user inserts a new parent of a node. Then the system tries to calculate which are the possible parents for the node on the basis of the definitions in the DTD. The parent constraints reduce the number of the possible choices. All children constraints check whether the children of a node are among a list of possible elements. The list of possible elements can be relative to the context of the node and in this case they are selected by an XPath expression. If the content of the node is textual, then it is first tokenized and then checked whether each token is in a set of tokens, defined by the constraint. The all children constraints cannot be used in information entering support mode. Some children constraints also impose restrictions over the children of a node, but instead of all children; in this case some of them are necessary to be members of the possible children, determined by the constraint. In the mode of user entering information the constraints show to the user a list of the possible values for a new child of the node and he/she has to choose one. The new value can be incorporated in any position of the node's content. Again, if the content is textual, then it is first tokenized. Similarly, some attribute constraints impose restrictions on the value of a node's attribute. The difference is that in this case the values have to be textual.

The three kinds of constraints, which can be used for

changing the content of the document, are exploitable also as rules. They become rules when they determine exactly one value on the basis of the context. In this case the user is not consulted and the value is entered automatically.

2.3.2. Regular Expression Constraints

In this kind of constraints the selection of nodes, which the constraints will be applied to, is defined by XPath expressions. The contents of the selected nodes must match a description given as a regular expression in the constraint. This kind of constraints work in validation mode. During application the selected nodes are split into two sets, containing nodes matching the regular expression and nodes which do not match. The user can navigate subsequently through any of these sets of nodes.

These constraints can be used for simulation of XML Schema constraints over textual nodes. In addition to checking the content, these constraints - via the XPath expression - can also determine the context of the elements, which they will be applied to. In this way they can be used for imposing regular constraints in addition to these in the DTD making them more specific on the basis of the surrounding context. This is very useful, for example, when someone compiles a dictionary. Usually the DTD defines some very general content model for the elements, but in a concrete lexical entry a more specific model is realized.

2.3.3. Number Restriction Constraints

Here again the selection is defined as an XPath expression. On the selected nodes separately another XPath expression is evaluated and the result from each evaluation is converted to a number using the rules defined in the XPath specification. A constraint is satisfied for a node if the corresponding numeric result is in a range given by two numbers MIN and MAX. The MIN and MAX values can be dynamically determined for each node by other two XPath expressions, which return numbers as results. The number constraints are very appropriate for stating general constraints over the whole document. Such constraints include arbitrary complex XPath expressions in their predicate part and otherwise they always select the root node if the predicate is satisfied in the document and require that there is exactly one such element.

2.4. Additional Tools

Another tool in the CLaRK System which can be used for retrieving information from text corpora is the Statistics tool. This tool is used for counting the number of occurrences of certain tokens or/and XML elements within documents. By XPath expression the user can specify the location and type of data she/he is interested in. If tokens are to be counted, then a tokenizer can be specified and further more only tokens of certain types can be counted. The information which is given for each token is: the token itself, its type, the number of occurrences and the percentage from all. The result can be stored as an XML document and used later for different purposes.

The Sort tool is used for reordering XML structures depending on certain criteria. The structures (nodes) to be sorted are selected by XPath expressions. The nodes are compared depending on some related features pointed by other XPath expressions. The sorting procedure can be performed in different layers, depending on the number of features to be compared. Thus, on the first stage nodes are sorted by the first feature definition (primary sorting). If any nodes are equal for this criterion, a secondary sorting can be performed on the basis of a second feature definition and so on. There are also some options concerning the feature comparison itself, i.e. sorting order, the treatment of capital and small letters, the treatment of whitespace symbols, tokenization and comparison by token types and others. There is also a possibility the text nodes to be compared in reverse order, i.e. instead of comparing letters from left to right, they are compared starting from the end. This tool can also be very useful for dictionary compilation. The dictionary entries can be grouped and sorted by any related feature, for example part-of-speech, including some more specific characteristics.

3. Rapid Prototyping

In the process of corpora creation the user needs to apply more than one of the above tools and some of them — even several times. Some of the applications and their order depend of the result of the previous tool. Thus in order to facilitate this kind of corpora processing we implemented a simple macro language able to control the applications of other tools in the system.

In the CLaRK System most of the tools support a mechanism for describing their settings. On the basis of these descriptions (called *queries*) a tool can be applied only by pointing to a certain description record. Each query contains the states of all settings and options which the corresponding tool has. In other words, each query has all the necessary information for applying the tool without any additional information or user interaction.

For user convenience and debugging purposes the queries themselves are represented in XML format. Within the system they can be treated like ordinary XML documents having their names and DTD assignments. For each kind of queries there is a special DTD included in the distribution package of the system. There the user can see the required structure for an XML document to serve as a query.

Each of the basic processors described so far works over a document and the result of this work is saved in the same document, or a new document is created.

Once having this kind of queries there is a special tool for combining and applying them in groups (macros). During application the queries are executed successively and the result from an application is an input for the next one. The final result is given by the last query application.

For a better control on the process of applying several queries in one we introduce several conditional operators. These operators can determine the next query for application depending on certain conditions. When a condition for such an operator is satisfied, the execution continues from a location defined in the operator. The mechanism for addressing queries is based on user defined labels. When a condition is not satisfied the operator is ignored and the process continues from the position following the operator. In this way constructions like `IF-THEN-ELSE` and

`WHILE-DO` easily can be expressed.

The system supports five types of control operators:

1. `IF (XPath)`: the condition is an XPath expression which is evaluated on the current working document. If the result is a non-empty node-set, non-empty string, positive number or true boolean value the condition is satisfied;

2. `IF NOT (XPath)`: the same kind of condition as the previous one but the approving result is negated;

3. `IF CHANGED`: the condition is satisfied if the preceding operation has changed the current working document or has produced a non-empty result document (depending on the operation);

4. `IF NOT CHANGED`: the condition is satisfied if either the previous operation did not change the working document or did not produce a non-empty result.

5. `GOTO`: unconditional changing the execution position.

Each macro defined in the system can have its own query and can be incorporated in another macro. In this way some limited form of subroutine can be implemented.

As it was mentioned above, each basic tool runs over one document and the result is stored either in the same document or in a different document. Sometimes it is necessary for the different queries in a macro to work over different documents. For instance, the extraction tool extracts some fragments in a new document and then a transformation is applied over the new document. In order to support this we have implemented two kinds of macros in the system. The first kind of macros work over a single document. The second kind of macros allows the different queries included in it to work with different set of documents. The last feature is very important in order one to be able to implement a prototypical processing over a whole corpus. This is why we consider CLaRK System as a good platform for a rapid prototyping: building a system that demonstrates a feasibility of some idea. In this way someone can construct new processing tools in the course of corpus creation.

4. Applications

At least two examples of the macro language usage will be demonstrated at the conference: partial parsing and treebank transformation.

The first example contains a special grammar for explication of the internal structure of maximal chunks: for instance, the `NP` groups recognition grammar for Bulgarian text. First, two cascaded grammars detect several kinds of maximal `NP` groups without an explicit internal structure. The third grammar detects the `NP`s' internal structure, i.e. embedded `NP`s in the maximal ones. Because of the recursive structure the last grammar has to be applied until it still detects new `NP`s. This is done by the usage of conditional operators.

Another interesting application of the macro language and the tools of the system is the treebank transformation (Ule 2003) for error spotting. The idea is to apply statistical measure over the distribution of context-free-like productions within context in order to recognize rare cases which can be errors in treebank. The implementation includes the following steps: application of the statistical tool for collecting data about the productions in the treebank and their contexts. Then the statistical measure is calculated for each production in general and in each specific context via XPath mathematical functions. The results are stored for each case as values of some attributes. Then the data is sorted and the productions which need transformation are determined. Then the actual transformation is done over the treebank. The whole process is then repeated several times depending on some criteria.

These two applications show the potential of the system for a rapid prototyping by combination of different processing modules in different processing strategies.

5. Conclusion

In this paper we presented the basic tools of the CLaRK System. Additionally, the macro language of the system is described. The macro language allows arrangement of the basic tools and macros in procedures. A set of control operators allows for changing the consequent order of application of the tools in the macro. All these advantages turn the CLaRK system into a powerful prototyping platform.

6. 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. pp 558–560.

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

Kiril Simov, Alexander Simov, Milen Kouylekov, Krassimira Ivanova. *CLaRK System: Construction of Treebanks.* In: Proc. of The First Workshop on Treebanks and Linguistic Theories (TLT2002), 20th and 21st September 2002, Sozopol, Bulgaria. pages 183-198.

Kiril Simov, Alexander Simov, Milen Kouylekov, Krasimira Ivanova, Ilko Grigorov, Hristo Ganev. *Development of Corpora within the CLaRK System: The BulTreeBank Project Experience.* In: Proc. of the Demo Sessions of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03), Budapest, Hungary. 2003.

Kiril Simov, Alexander Simov, Milen Kouylekov. *Constraints for Corpora Development and Validation.* In: Proc. of the Corpus Linguistics 2003 Conference, pages: 698-705.

Tylman Ule. 2003. *Directed Treebank Refinement for PCFG Parsing.* In: *Proc. of The Second Workshop on Treebanks and Linguistic Theories (TLT03)*. Växjö, Sweden.