XGate and XRG: tools for visually editing, querying and benchmarking XML linguistic annotations.

Francesco Cutugno[1], *Leandro D'Anna* [2]

[1] Department of Physics - NLP Group, University 'Federico II' of Napoli, Italia
[2] Department of Linguistics and Literature, University of Salerno, Italia
cutugno@na.infn.it, ldanna@unisa.it

Abstract

Presently there is a great request in many fields of corpus linguistics of manually annotated texts and transcriptions. XML is rapidly become the principal instrument for linguistic markup even if in many occasions people operating in this field, mainly linguists, are not really experts in managing this technology. Although, at least in principle, many tools are available on the Internet, most of them are not easy-to-use and/or free of charge.
This work presents a set of software tools supporting the activity of producing XML data files with a special attention to linguistic annotation. Two products will be described in details: the first one, XGate, is a program which supports editing and querying of XML files and at the same time implements a semi-automatic method to synchronize such files to a modified DTD or Schema. XML file editing is performed visually, the document is showed in its tree-like form, tags and attributed are filled by the user as in a form. Querying is realized using a further visual interface that implements most of the XPath syntax graphically, furthermore query result can be again queried by means of a tool that automatically analyzes the structure of the former results.
The second program, XRG, is a tool for XML native database benchmark. XRG produces XML files of a given complexity in terms of node numbers, levels of direct and indirect recursion, horizontal width and vertical depth of the tree. Generated file can be queried with a built-in XQuery module and further statistical analyses are possible. If you have an XML DBMS and you want to verify performances, you can use XRG to generate a 'certified' dataset with which to evaluate your system.
Both tool are directed to non-experts, users are not asked to know XML and can visually manipulate their data in most of the software sections. Powerful and user-friendly interfaces have been developed for this aim. XGate and XRG are open software tools and it is possible to download executables (Windows + .NET framework platform only) and source codes for free from the portal of the Italian project 'Parlare Italiano' [1].

Index Terms: XML editing, XML querying, visual interfaces, benchmark.

1. Introduction

In many fields of computational linguistics there is a growing request of corpora including annotations, labeling and markup. At the same time XML is rapidly becoming a standard for the most used procedures for introducing markup to texts and transcriptions. Starting from the efforts made within the TEI framework [2] and following the developments produced in various international projects (as for example TIGER, NITE, AMI [3,4,5]) the necessity to manually add labels and annotations to corpora using XML is growing up even if it is not rare at all that people working in this field still misses specific skills for the use of this technology. There is a large request of tools for XML files managing having user-friendly interfaces; these tools are mostly used within the frame of public research, where economical resources are usually scarce and in which investment on software production cannot be a relevant portion of the overall budget. Another crucial question under discussion in the recent years is how to retrieve information directly from dataset formatted in XML as in many cases these kind of data cannot easily transferred into a relational database management system. Querying and searching into an XML file has been made possible, but not easy, by using specific query languages recently proposed by the W3C consortium. We present here two XML open source tools, XGate and XRG, conceived and projected having in mind some precise constraint: powerful user interfaces both for editing and querying, robustness against errors introduced by the user, support to guided recovery procedures in case of redefinition of the document structure, possibility of hard technical benchmarking both for data and for querying. In the spirit of open source software, these programs are freely downloadable. They run under Windows and require the Framework .NET 2.0.

2. XGate

XGate is a tool born to help linguists to manage with syntactic treebanks as produced during the process of texts manual annotation having as final deliverables XML files. It was conceived to solve all the problems occurring when managing with semistructured data as linguistic treebanks or, in general, with every data represented by a tree and saved in XML. Then XGate can be seen as a simple, efficient and intuitive tool to edit and query XML files. It is divided into two main modules plus some additional feature supporting and facilitating the annotation process. The two main sections are: Editor and Query Manager, while the additional features are a wizard for resyncronizing XML documents when DTD changes according to variations required by the user, an XML-Schema Manager, a tool for deriving a valid DTD from an XML document and a Notepad.

2.1. XGate: Editor

XML files are not easy to read for a non expert user. XGate proposed a visual interface into which the XML document is showed as a hierarchical tree. Our program analyses the DTD related to the document to be created and uses this knowledge to directly support the annotation process suggesting relations between tags and attribute values when these are to be choosen in a closed set.

The interface presents many frames in which all knowledge required for texts annotation is included, it is projected in such a way that the user is accmpanied step-by-step during the annotation process, XML syntax is verified on the fly and validation process takes place every time the file is saved.
In fig. 1 the Editor interface is showed.

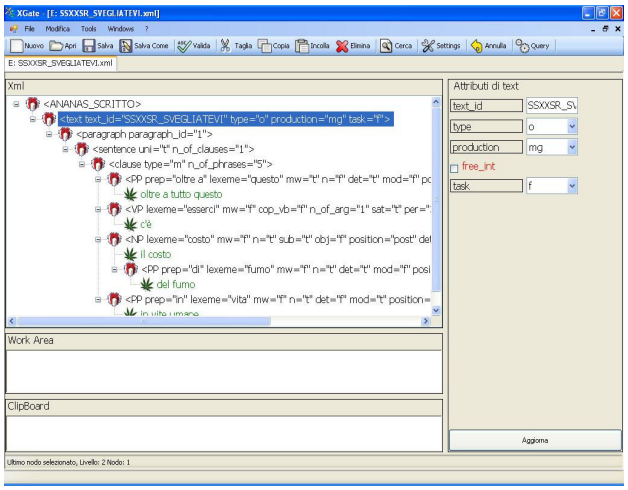

Figure 1*: XGate: Editor Interface.*

2.2. Query manager

Querying Treebanks in particular and XML native database in general, is a complex task. In many case query results are conditioned by the level of articulation that the hierarchy of tags in the document imposes, while at the same time, closed loops and recursion - typical in an XML document related to liguistic annotations - make the information retrieval process ambiguous and depending on the path choosen to visit the database.
To partly recover this problem W3C proposes two standards for querying XML documents: XPath and Xquery [6,7]. The first defines a query language based on paths on the tree structure. User defines a template for a possible sequence path in the documents with well defined constraints on the tags and the attributes, specifies the class of possible entry points in the document from where the search process can start and the query returns all the nodes or the subtrees satisfying the request.
XQuery adds to XPath the usual programming statements as if-then-else, loops, variable definition and assignment in order to improve possibilities offered by the language for querying. XGate implements an user friendly interface to produce single XPath queries. According to the analysis of the user requirements it has been decided not to implement XQuery in this tool as it would have required the user possess at least basic programming skills. Luckyly, most of the user requests can in any case reconducted to a sequence of XPath expressions. XGate implements a visually guided query construction, user does not need to know XPath syntax and is guided through the process of retriving information on the base of a preanalysis of the document under examination of the related DTD.
Fig. 2 shows an example of the Query interface.

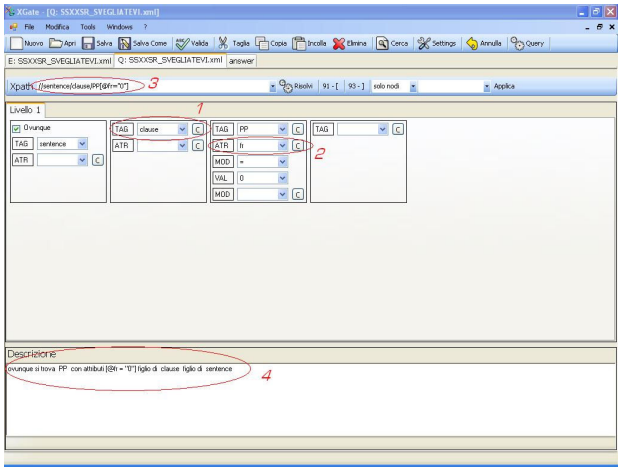

Figure 2: *XGate: Query interface.*

The output from the query is automatically formatted in a new XML document. The user can, at this point, formulate a new query on this output making once again use of the interface, XGate automatically determines a possible DTD for this new document (this process is obviously mandatory if you want to reuse the interface), and allows to restart the querying process. In this way it is easy to increase the level of data selection capacity even not using XQuery that would have been the preferred choice in case of an user with programming skills.

2.3. XGate - Additional features

2.3.1. XML to DTD

As formerly described, within XGate it is often necessary to indirectly obtain the structure of the XML document under examination. This can be made making use of a tool that finds a possible DTD for a given XML document. As it is well known, this process is not unique as a single XML document can be related to an infinite class of DTDs depending on some factors like undeclared tag dependencies, or implied attributes role, closed vs open set values for attributes and so on. Our tool produces instances of possible DTDs according to constraints defined by the user and mainly balanced to permit most of the automatic analyses performed by the system on the XML documents.

2.3.2. Notepad

It is in any moment possible to visualize the textual aspect of the XML document using an internal notepad. This tool is also useful to pre-load a text or a transcription to be annotated.

2.3.3. Statistics on docs

A very useful tool for linguists permits listing and counting of all tags, attribute and values encountered within the XML document loaded into the interface.

2.3.4. XSL editor

We included a third party open source product into XGate, namely XSDEditor [8]. This tools permits the visual organization of the tagset and of attributes and is really useful in the phase of database project.

2.3.5. XML - DTD re-synchronizing Wizard

Linguists often change the metadata structure during the course of annotation if they encounter phenomena that were not foreseen. In this case the already made work is to be re-controlled and eventually corrected. At the same time XGate automatically verifies files wellformedness and checks validity against the assigned DTD. When a file is not valid both because of an error in the annotation or because of a structural (and pre-ordered) change in the DTD, it is possible to modify it in order to be a valid file. When possible the modifications are realized automatically without requiring a user feedback. If this process finishes and the document is not valid, the module begins an assisted correction in order to satisfy the DTD file asking the user to choose changes to be made in the file in every miscorresponding tree node.

3. XRG

XML native databases are characterized by the presence of closed loops and recursion. Under these conditions a given query on the data, differently from what happens with relational database, can produce different outputs depending on the entry point in the document and on the level of recursion that is present in the semistructured dataset.
In this view it is really important, in order to evaluate the real power of expressivity of your queries to test them on a certified dataset before testing them on real data.
Xml recursive generator (XRG) is an automatic tool for creating Xml files using a pseudo random generation. The Xml files is generated by XRG in a controlled mode starting from a given DTD file and setting the level of document 'complexity' assigning values to a set of parameters by means of a very easy visual interface. This process can be divided in two stage: a first stage in which a deep analysis of DTD is performed and a second stage where the generation of XML files takes place together with the possibility to evaluate and compare difference among different runs on the same DTD and with the same number of tag. Finally it is possible to query the generated XML file using the standard W3C query language Xquery.

3.1. Pre-analysis stage

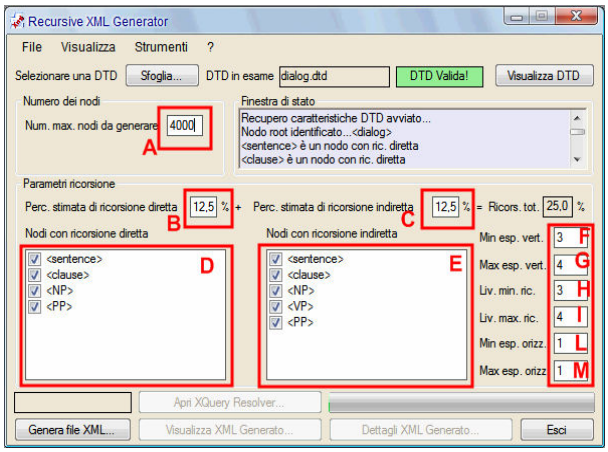

Figure 3: XRG *GUI. In rectangles forms to set parameters.*

First of all the DTD file was checked to verify its validity. Then all possible tag elements in DTD are analyzed to determine if they have a simple or recursive structure.

For those that have a recursive structure, it is determined if they have a direct recursion (eg. TAG1 can include TAG1 between possible sons) or indirect recursion (eg. TAG1 can have a TAG1 as possible nephew or successive descendants). It is important to notice that a single tag can have both direct and indirect recursion.
By mean of a user friendly GUI (see rectangles in fig. 3) it is possible to choose these parameters:

A. The overall number of elements that would be insert in XML file;
B. Maximum percentage of tags with direct recursion;
C. Maximum percentage of tags with indirect recursion;
D. Which tags from the list of element with direct recursion to use in generation;
E. Which tags from the list of element with indirect recursion to use in generation;
F. The minimum height (vertically) of XML file seen as a tree;
G. The maximum height (vertically) of tree;
H. The minimum depth where it is possible insert recursive elements;
I. The maximum depth where it is possible insert recursive elements;
J. The minimum width (horizontally) of tree;
K. The maximum width (horizontally) of tree;

During the parameters setting operation, an automatic check discards value erroneously inserted.

3.2. Generation stage

The generation engine is based on a sequential algorithm that controls step by step that the partially generated tree satisfies all structural constraints imposed by DTD.
This approach permits to obtain good performance both in time and in number of elements even on computers with poor performance. Obviously the generation time increase whit the number of elements in the tree with an experimental exponential trend for our module. It is interesting to note that noticeable changes in the structure of tree without modify the overall number of elements, causes little change in the generation time.
XRG has been investigated even in terms of temporal complexity in relation to the number of generated XML nodes. In fig.5 such performances are showed.

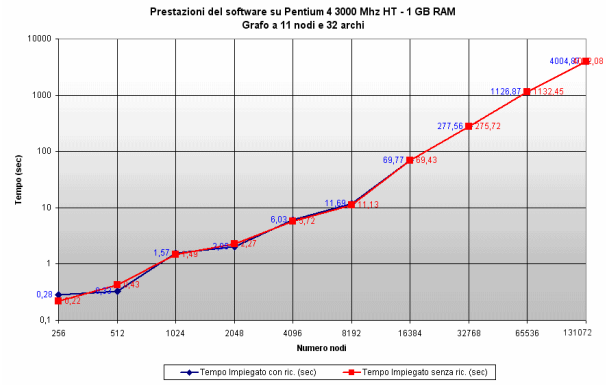

Fig.5: *temporal complexity for XRG (double log scale, xaxis: number of generated node, y-axis: sec)*

After a quick generation (e.g. 16384 elements in about 70s with a CPU PENTIUM4 3000Mhz and 1 Gbyte), it is possible save the XML file generated and make accurate evaluation of the output.

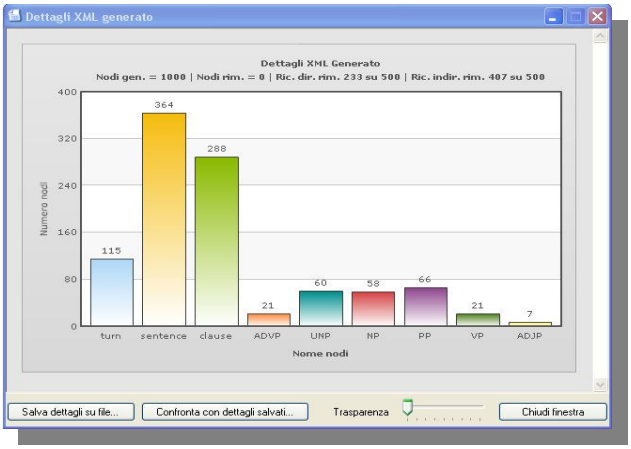

Figure 4 XRG*: Graphical evaluation of output.*

To do this a graphical tool for statistics (see fig. 4) permits to analyze the distribution of the tags in the XML file.
All the statistics can be saved both as text and as image. It can be useful to make further comparisons between two file generated with the same parameter set.
Finally, in order to realize more complex analysis on the XML file, it is possible to use a query module that permits to realize XPath 2.0 and XQuery 1.0 queries and it is based on the Saxon processor [9].

4. Cross controlling

To evaluate the correctness of the generated XML file and the related statistics computed by XRG, we have used the query module of XGate.
These cross controls can be conceived in two ways:

1) It is possible to check if percentages of direct and indirect recursion were satisfied into the generated XML document.
In order to do this we have imposed for parameters B) and C) (see figure 3) the same value chosen as higher as possible in relation to the constraints derived by the assigned DTD. Furthermore we have assigned all the recursion to only one tag for every type of recursion.
Then using a simple XPath instruction like: **//TAG1/TAG1** we have verified the overall number of direct recursions.
In order to verify the number of indirect recursions, we used the XPath instruction: **//TAG2/*/TAG2**

2) with the second test we checked if the XML files generated with XRG were correctly created according to the tree-like structure depicted in the document design made by means of the DTD. For this aim we assigned the same percentage for direct and indirect recursion using only one tag for every type of recursion. Then we have imposed that the positive value of parameters H) is 'value1'.
To check that Xml files respect this ties we use these XPath instructions:

//TAG1[@deep<value1] for direct recursion
and //TAG2[@deep<value1] for indirect recursion.

Many other check controls can be made changing opportunely parameters in XRG and using the power of expressivity of the XPath commands.
In both cases requirements expressed in XRG perfectly match with the verification made by XGate.

5. Conclusions

The tools have been produced within a national project (see next section) within which an observatory on researches regarding spoken Italian has been realized. A web portal has been designed and data, tools, corpora and other resources regarding Italian language were collected. Within this project, the development of XGate and XRG has been realized in conjunction with a group of users ranging from more to less XML experts, their feedback has been constantly taken into consideration in all phases of the project. XGate has been tested on two different corpora: a syntactic treebank, ANANAS and a corpus describing dialogue pragmatics, PraTID [both available in 1]. The first corpus presented an high level of complexity due to the high number of attributes for each tag and to the presence of direct and indirect recursion in many levels of the document structure; furthermore , during the construction of the corpus, authors decided in various occasions to modify the DTD to introduce labels and to describe phenomena not initially foreseen. PRaTID was made with a simpler XML structure, for this corpus times for labelling were evaluated. Results reports an high grade of satisfaction by all the users. As already seen in the previous paragraph, XRG has been widely tested for benchmark in cross validation with the XGate query generator section.

6. Acknowledgements

This work has been funded by the Project 'Parlare Italiano' (PRIN 04 and PRIN 06 -MIUR Italy, National coordinator Miriam Voghera). LDA and FC have projected the tools and have defined the overall software architecture and fixed requirements and constraints. XGate has been written by Gennaro Cavezza, Emilio Diana and Antonio Vuolo. XRG has been written by Raffaele Liguoro. Miriam Voghera, Renata Savy, Giusy Turco, Simona De Leo hardly tested the programs. Annamaria Landolfi and Carmela Sammarco wrote manuals and docs.

7. References

[1] Parlare Italiano: http://www.parlaritaliano.it
[2] TEI: http://www.tei-c.org.
[3] The TIGER corpus: http://www.ims.uni-stuttgart.de/ /projekte/TIGER/TIGERCorpus
[4] J. Carletta, S. Evert, U. Heid and J. Kilgour, "The NITE XML Toolkit: Data Model and Query Language", Language Resources and Evaluation, 39(4):313-334, 2005. http://nite.nis.sdu.dk/
[5] AMI: http://www.amiproject.org/
[6] Xml path language (Xpath) 2.0 W3C recommendation: http://www.w3.org/TR/xpath20/
[7] Xquery 1.0 XML query language W3C recommendation: http://www.w3.org/TR/2007/REC-xquery-20070123
[8] XDSEditor: http://www.codeproject.com/dotnet/xsdeditor.asp
[9] Saxon XQuery processor: http://saxon.sourceforge.net