Andrea Bolioli, Luca Dini, Pietro Mercatali and Francesco Romano, 'Models of "Novelle" and Normative Grammar' in T. Gordon (ed.), *Legal Knowledge and Information Systems. Jurix 2004: The Seventeenth Annual Conference*. Amsterdam: IOS Press, 2004, pp. 39-48.

Models of "Novelle" and Normative Grammar

Andrea Bolioli[1], Luca Dini[1], Pietro Mercatali[2] and Francesco Romano[2]
[1]*CELI S.r.l. Turin Italy*
[2]*ITTIG-CNR, Via Panciatichi 56/16 - 50127 - Florence, Italy*
{mercatali, romano}@ittig.cnr.it

Abstract. In this paper we will present a method for mining information within legal texts, in particular in regards to corpora of statutes. Text mining, or more in general Information Extraction, can provide valuable help to people involved in research about the linguistic structure of statutes, and, as a side effect can be the seed for a new generation of applications for validation and conversion in the legislative domain.

1 Scope and assumptions

For the communication of legislative sources through the Internet, the parliamentary and governmental institutions of many countries[1] have begun a process of converting their "deposits" of these into a standard format for facilitating the retrieval and display of texts.

The XML mark-up language seems to be the tool able to constitute the standard for the treatment and the web publication of the legislative information [1-2].

In order to adopt this language as a effective and really shared standard, two factors, in our opinion, must interact.

A) Definition and promotion of a "controlled" legislative language

At the cost of seeming superfluous and obvious, it is worth repeating that the promotion and dissemination of legistic standards[2], and, therefore, of a strongly binding normative language can only facilitate the adhesion and use by bodies drafting norms to the technical standard proposed with the XML language. These drafting rules have been applied and complied with in the drafting of legislative instruments enacted by the State and Regions since the end of the 1980's. After the 1980's, these rules became increasingly wide-spread and put into application even if, on leafing through the legislative documents, it cannot be said that this application was always rigorous and uniform [4].

B) Use of tools for natural language recognition

It is evident that the presence of common rules consolidate the definition of text models.

It is also evident that this modelling assists in the use of Natural Language Processing (NLP) means for the recognition of the structures of legislative instruments and their tagging according to the XLM standard. In fact, this tagging will be difficult to obtain from the law-maker as it is extraneous to the tasks and objectives involved in his normal activities. If other professionals do it later, it provoke (and in some cases it has already provoked) an often unsustainable increase in the time needed and the costs involved in building and managing the legislative knowledge base, structured according to XLM standards.

[1]See, for example: the data bank of legislative instruments of the Australian State of Tasmania at http://www.thelaw.tas.gov.au and the DTD relating to the parliamentary acts in the United States on the site http://xml.house.gov.

[2]Circular 20 April 2001, no. 10888 of the Presidency of the Council of Ministers, "Regole e raccomandazioni per la formulazione tecnica dei testi legislativi". For an in-depth illustration of the rules for legislative drafting in Italy and in Europe see [3].

2 The project

It is within the perspective of the implementation of a parsing system efficient for the automated recognition of the structures of legislative instruments and the subsequent tagging and conversion of these texts in XML format that we shall now begin the description of the research presented here.

2.1 Definition of models like set of legal rules

The application of a device for parsing, requires a set of rules to be written for the identification, in the texts, of linguistic structures which are bearers of the information we wish to extract. We can call this the compilation of a specific grammar of the domain or of the corpus of the texts to be analysed. The grammar is made up of a set of models defining the linguistic structures; in turn, the models include one or more rules representing a linguistic structure which are subsequently compiled according to the syntax of the parser for the text analysis and the information extraction. Computational linguistics indicates the extraction of the set of rules and models from the corpus of the texts directly linked to the domain to be dealt with as the most efficient method for constructing the grammar [5]. In other words, we are trying to reconstruct rules and models a posteriori, extracting examples of linguistic structures. The application of this methodology to legislative instruments, however, overlooks the specificity of the nature and function of these texts. The legislative instrument has, by definition, a prescriptive function, or in other words, it influences the behaviour and status of the addressee, who cannot escape from this function. In virtue of this, the request (it is based on the same principle of representative democracy which legitimates and, at the same time, binds those who draft legislative acts, namely, the legislator) is that the legislative instrument responds to a set of rules that dominate and, at the same time, stand beside, integrate, and sometimes modify the rules that make up natural language (in our case Italian).These rules can be defined as legal rules, The category is broad and varied; it includes within it rules with prescriptive force, that vary strongly based on the source from which they come, to whom they are the addressees, the sanctions they bring with them, etc..

2.2 Identification and description of the textual structures corresponding to the defined legal models

For the implementation of the grammar that will then be utilised by the parser, it is necessary to integrate the models extracted from the legal rules with the linguistic rules.

Here the following three clarifications need to be made:

1. more than integrate, we should talk about the translation of legal rules into linguistic rules, or in other words, that the legal rules are bound or expressed by linguistic rules even if both assist and integrate in the construction of the model able to be implemented;
2. some legal rules, especially some legistic rules, because they regulate the drafting of the legislative text, originate as linguistic rules and the law does nothing but receive them, reinforcing their cogency;
3. from what has been said, both the linguistic rules received by the law, and the linguistic rules that transmit legal rules have, or in some measure receive, the prescriptive force of the law and, therefore, we can talk about legal language as a strongly bound language.

We can say, for example, that the legal rules prescribe that a type of amending provision, is manifested through the action of substituting parts of the text; the linguistic rule of synonymy

enables us to say that the action of substitution is expressed through verbs: to substitute, to change, to alter, etc... In this case, the linguistic rule goes to integrate itself with the legal rules, in the construction of an efficient model for the purpose of the function of parsing the text. The integration may, however, also concern cases where the legal rule is not so much integrated but the grammar of the legislative instrument is completed, going to describe the linguistic structures, to which the legal rules do not correspond[3]. From textual practices[4], we can extract rules that go to form other models of linguistic structures found in legislative instruments (or to integrate those models obtained with the legal rules). These are models extracted on exemplifying bases, resorting to the analysis of texts according to the methodology practised by computational linguistics. Clearly, these models do not have the same precision and, above all, the same prescriptive force of those constructed on the basis of legal rules. We have, therefore, only used and implemented them in the parser as a residual category. We call these last models "malformed" models. This is not so much because they are not correct from the linguistic point of view, but to contrast them with those corresponding to the legal rules, indicated in the previous paragraph, which we define as "well-formed". Nevertheless, legislative instruments contain linguistic structures that do not correspond to the models described, because the texts may contain actual legal and linguistic errors. In these cases, we complete the grammar to be implemented in the parser, with rules (we define them as "case-based" rules) which represent exceptions to the models: they are actual errors or exceptions to the rule, just like exceptions in linguistic grammars.

2.3 Choice of the technical tool for the implementation of the parser and for information extraction

The suitable tool for the recognition and tagging of a legislative instrument has been identified in the Sophia 2.1 system of parsing which uses the methodology applied to finite state automata and the finite state transducer: this is software which is flexible and configurable and which enables rules and specific models (already defined or in the course of definition) to be formalised.

In particular, we are working with this software on analysing and tagging the first sample of legislative instruments in the following phases:

- normalisation of the entry text, properly tagging all those structures and textual segments that can be recognised on the basis of characters;
- lexical (syntactical category) and morphological (flexion passages) analysis of the text in input;
- disambiguation of the syntactical category of the words (*Part of Speech Tagging*);
- partial syntactical analysis (called *chunking*), aimed at identifying the minimum syntactical groups;
- semantic analysis and identification of the relevant conceptual structures in the text in input;
- conversion of the analysed document from the original format into the XML format.

[3]For example, for the *novella*, the legislative drafting rules provide for well identified functions (integration, repeal, substitution) to each of which one or more linguistic structures correspond which are also carefully described in the law-making rules. We can also find *novelle* with a replacing function in legislative instruments: they arrange the repositioning of a part of the text from one point to another of the article. The replacing *novella* is not, however provided for or regulated by the rules on law-making.

[4]The term *practices* is used here in accordance with its common meaning of recurrent behaviour (in our case, the behaviour of recurrent drafting) and not in the technical legal sense that classifies them among the sources of the law.

2.4 Compilation of the grammar and the automated analyses of the sample

The compilation of the grammar in the syntax of the chosen parser takes place by using the Workbench of the system described in paragraph 2.1., through drafting legislative rules, that formalise the defined models and permit the automated identification of the described linguistic structures and the information extraction.

The choice of the sample of legislative instruments to be analysed must obviously respond to the representative criteria of the legislative linguistic domain, from which we intend to extract the information. In describing our research, we indicate the following criteria for identifying the sample which is the object of this initial analysis.

The analysis of the sample can then be carried out in subsequent phases in order to:

- evaluate the results obtained;
- integrate and modify the well-formed models defined *a priori*;
- identify and formalise case-based malformed models and rules;
- extend the analysis to a gradually widening *corpus* to verify the efficiency of the parsing system.

3 Initial analysis

We have decided to experiment the method described for the automated recognition and extraction of three typical structures of legislative instruments, structures representing: legislative delegation, express textual amendment or *novella,* express external textual reference.

In the following paragraphs, we shall present the modelisation of the textual structure of the *novella,* on the basis of the normative and linguistic rules and an initial formalisation of those rules in the syntax of the parser whilst, for the initial analyses carried out on the structures of the legislative delegation and on the system for recognising legislative referrals, we refer you to our earlier works [6-7].

4 Express text amendment or novella

4.1 Definition of the structure and the constituent elements

Amendment provisions, according to Sartor, fall within the main types of legislative links, classified on the basis of their impact on the legal provision involved. Amendments distinguished from the other large branch of referrals or references, are legislative links characterised by the fact that the active provision affects the passive provision, eliminating it, changing the text or changing the legal significance (whilst leaving the text unchanged). This effect is, instead, lacking in the referral, where the active provision avails itself of the passive provision to complete its meaning, without influencing the latter [8].

In relation to the nature of the impact of the amendment of the provision on the passive provision, we distinguish between textual amendments (that amend the text of the passive provision), time-based amendments (that influence the period of time of the applicability of the passive provision), material amendments, (that amend the legislative content of the passive provision without affecting the text). We shall only look at the first type, the express amendments of the text which, traditionally, lawyers in Italy call the *novelle*.

Indeed, it is perhaps more correct to say that the function of express legislative amendment is expressed through the following three aspects:

- the structure of the novella, made up of an introductory part, called *subparagraph*[5] and a part that contains the *express textual amendment.*
- the characteristics of the amending legislative act and the amended act: indispensable for subsequently being able to reconstruct the amending links between the different legislative sources;
- the citation with which the document to be modified is cited, that expresses the legislative reference (also a textual reference), a fundamental element of the amending provisions.

On the basis of the three aspects mentioned here, we have endeavoured to define and describe the qualifying elements of the amendment provision. The description, which is set out here, is derived from the rules for legislative drafting and confirmed from the analysis of some State legislative instruments. Once the elements making up the amending provision had been identified and described, we were able to propose a classification based on two of the elements we believed to be particularly important: the action of amending and its object [9]. In particular, on the basis of the action of amending, a distinction can be made among the following: repeal, integration and substitution. As far as the object is concerned, the amendment, instead, operates on either a part (supra-part, article, paragraph, etc.) or on a part of the legislative discourse. It is obvious that each of the identified actions can operate on both the object *"part"*, and on the object *"part of the discourse"*.

4.2 Formalisation of the rules and implementation of the parser

Each of the types of amendments identified, on the basis of the given classification, was formalised in the syntax of the parser we used. In this way, we got a set of recognition and extraction rules of the "well-formed" amending provisions, based on the models extracted from the Italian regional rules on legislative drafting.

As we mentioned earlier in the introductory part, once the "well-formed" models of amendment were implemented, we moved on to an analysis of a corpus of legislative texts with a dual scope in mind:

- to verify the validity and flexibility of the formalised rules for the recognition and extraction of the amending provisions, also in cases where the linguistic structure used in the text is not exactly the same as the given model;
- to identify the structures of amending provisions which, although they are logically, legally and linguistically correct, cannot be reduced to formalised models.

In writing the rules which implemented the amendment model we mainly used three of the eight modules of Workbench of Sophia 2.1: the Compounder, the Lexical Semantics and the Semantics.

In the compounder module, we defined the complex prepositions and the adverbs important for the purpose of identifying, within the part, the position where the amendment will act (the final, the last, at the end, before, etc.).

Then, each of these was attributed a semantic category (INIZIO [At the beginning], TRA [Between], FINE [At the end]) corresponding to the type of function performed by the syntagm within the amending provision[6].

[5] Understood as the 'part of the provision that introduces the amendment': it contains the purview aimed at specifying the relationship (substitution or integration or abrogation) between the provision in force previously and that provided by the textual amendment. The new paragraph generally ends with a colon, followed by the textual amendment placed between inverted commas.

[6] For example, to describe an amending provision of this kind:

In the second module (LexSem), we, instead, attributed to the verbal voices, with the various synonyms, the semantic categories corresponding to the various actions of amending (to substitute, to repeal, to insert, to add, etc.). It is in the semantics module where it is possible to write the necessary rules for the semantic analysis and for the identification of the significant conceptual structures in the input text. In fact, in this module, it is possible to build the rules that make up the models to be extracted. The pattern of this module will be made up of the previously assigned semantic categories. Naturally, a variable must correspond to every value attributed to the semantic category which is indispensable in the case where we wish to extract an important datum and therefore to create a template or also where we want to identify the beginning and end of the XML tagging, as in the case of the amendments.

For example the amending structure indicated in:

> *To <rif ...> Article 7 of Law No. 229 of 20 June 1988 </rif>, the following paragraph was added, at the end, " (text of the paragraph)".*

is translated into the following pattern of the Sophia parser:

> (IN:startm|A_PREP:startm)+DETX?+RIFB+[M_ALLCAT]*+RIFE:endpos+
> PUNCTX?+([E-SUCCESSIVE-MODIFIC-M]|[COME-MODIFICATO-M])?+
> PUNCTX?+(**INSERIMENTO** :vazione| **AGGIUNTA** :vazione)+
> PUNCTX?+FINE:endpos+PUNCTX?+(**COMMA_I** :vnov| **NUMERO_I** :vnov| **LETTERA_I** :vnov| **FRASE** :vnov| **STRINGA** :vnov)+ (PUNCTX)?+ **VIRGOLETTE** :startnov+
> [**M_ALLCAT-PLUS-RIF**]*+ **VIRGOLETTE** :endnov

As we can easily see, the amending action of integration was formalised in the semantic categories AGGIUNTA [Addition] and INSERIMENTO [Insertion], which we verified as being the sub-actions into which the action of integration can be subdivided.

Apart from the PUNCTX category which identifies the punctuation found in the discourse, and the semantic category (FINE) [End] which identifies the position where the amendment will operate discussed earlier, the further categories (VIRGOLETTE) [Inverted commas] identify, in a reasonably intuitive way, the inverted commas that delimit the *novella*, whilst the macro [M_ALLCAT-PLUS-RIF]* identifies everything to be found between the inverted commas.

Instead, the categories (COMMA_I, NUMERO_I, LETTERA_I, FRASE, STRINGA) [Paragraph_I, Number_I, Letter_I, Phrase, String] indicate the partition or the string that goes to integrate the amended provision. Two tags have also been introduced which identify the beginning and end of the reference (REFB and REFE). As we have already mentioned, the fundamental element of the amending provisions is the explicit legislative text reference with which the document to be amended is referred. The rules of recognition and extraction of the explicit text references have been implemented in a prior phase of the Project[7], and have been used to define the methodology we have already described and which we are currently perfecting and applying in the formalisation of the amending provisions. These tags are, therefore, only used to identify that portion of the text which contains the reference which is recognised and extracted by a different set of rules.

Before'<rif ..> Article 1 of Law No. 41 of 28 February 1986, </rif> , the following was inserted: "Art. 1"
The corresponding pattern in the syntax of the parser will be the one set out below:
INIZIO:vpred+DETX?+RIFB+[M_ALLCAT]*+RIFE:endpos+PUNCTX?+[COME-MODIFICATO-M]?+PUNCTX?+(INSERIMENTO:vazione|AGGIUNTA:vazione)+((DETX+AX)|ARTICOLO_I:endnov|STRINGA)+ (PUNCTX)?+VIRGOLETTE:startnov+[M_ALLCAT-PLUS-RIF]*+VIRGOLETTE:endnov

[7] For the description of this phase of the project, see [7]. It is worth noting here that the initial analyses carried out on laws passed in the 1990s making up part of the selected legislative corpus, confirm, for now, that, thanks to the models of regular citations compiled in accordance with the parser's grammar it was possible to identify and extract over 95% of the explicit legislative text references, conforming to legislative drafting rules.

We intend in this way to obtain a modular formalisation of the text structures, making it possible for every module to integrate, in the phase of the recognition and extraction of the information, with the already created modules with the objective of building an actual grammar that will enable a large number of segments and gradually more and more segments of the legislative discourse to be recognised.

An initial analysis of the text in input will, therefore, allow us to identify the reference found in the amendment and to tag the beginning and end, whilst a second analysis of the same text, will identify the structure to be amended. As already noted, the semantic categories REFB and REFE will enable the parser to identify the beginning and end of the reference found in the amendment. Therefore, the working hypothesis we are proposing provides for the formalisation of other legislative structures in the syntax of the parser and for integrating the various modules obtained, according to the system we have just described for the amendment and the reference.

In extending this methodology to other parts of the legislative discourse we plan to proceed in such a way as to permit immediate applications which leave aside the construction of a "universal legislative grammar" that, obviously, can only be a long-term objective. For example, the automated formalisation and recognition of references has led to a useful application for building automated links among legislative measures stored in a database or simply available on the Internet [10-11].

4.3 Results of the analysis carried out with the parser

We set out to analyse a sample of 16 laws chosen among State and Regional laws with the parser in a varying space of time between 1968 and 2004 .The initial quantitative results show that, of the 295 amending provisions found in the legislative texts, 161 amendments (namely, 54.6% of the sample) were correctly identified by the parser with the rules formulated according to the regional rules for legislative drafting. Other amendments (16, equal to 5.4%) were identified but with some inaccuracies (for example, a part of the text was inserted in the amendment which really did not belong to the provision) whilst 118 amendments (40%) were not uncovered by the system.

4.3.1 Analyses and modelisation of the non recognised structures

It was possible to single out the following categories with an initial analysis of the amending provisions which the parser was unable to identify: 1) multiple amendments, 2) multiple amendments of strings without main introductory paragraphs, 3) repetitive amendments, 4) conditioned amendments, 5) relocation.

1) *Multiple amendments.* The first type of amendment can be traced back to a structure in which, after the introductory part of the "*alinea*" or *introductory paragraph* where the amending action is generically introduced, the parts containing the *explicit textual amendments* are repeated as many often as the partitions which are amended are of the act modified by *novelle*. Each of these parts is introduced by a preposition, which we call *sub-introductory paragraph*. Each sub-introductory paragraph specifies the amending action and the individual sub-partition to be amended within the main introductory paragraph. We can distinguish multiple amendments according to whether the amending action in the sub-introductory paragraph is always the same or whether it is different. In the sub-introductory paragraphs, the reference is quoted in an elliptical way, in other words, the name, date and number of the act to be amended is not repeated but are only indicated the first time in the main introductory paragraph. It will be necessary to recognise every sub-partition modified by *novelle*, something which we have already seen to, by introducing special rules in the parser. It will, however, be indispensable to also provide a procedure which, after having recognised the

amendment as a multiple one, creates a correspondence between the elliptical reference to the individual sub-partition and the complete reference found in the main introductory paragraph.

2) *Multiple amendments of strings without main introductory paragraphs.* In this category, we have inserted a particular amendment which we found in the selected sample. The particularity was due to the introductory paragraph not being clearly distinguished by the *novella* and, above all, to the non explanation, in the introductory paragraph itself, of the verb “to amend”. The following is an example of this:

> *In Article 56, third paragraph, of the consolidation of the provisions concerning the statute of State civil servants, approved with Decree of the President of the Republic No. 3 of 10 January 1957, the word:*
>
> *"heard” [in the plural form] is substituted by the following: "heard" [in the singular form]; the words: "and the board of directors" are deleted.*

These are cases where, in our opinion, the amendment acts according to the followingschema: SUBSTITUTION + ABROGATION, SUBSTITUTION + INTEGRATION, ABROGATION + INTEGRATION, ABROGATION + INTEGRATION + SUBSTITUTION, ...

Obviously, the order and the combinations according to which the various amending actions can operate may vary with respect to the given schema. This case should only arise for the amendments of strings and not partitions.

3) *Repetitive amendments*. This is the case where the amendment acts by amending a word or a phrase every time these occur in the text modified by *novelle*.

4) *Conditioned* a*mendments.* We have also discovered possible and optional elements that may complete the amending provision such as: conditions about temporal effectiveness or another type of the amendment.

> *The provisions found in Art. 27, paragraph one, no. 9) and, restricted to references to information technology, no. 3), of Law No. 93 of 29 March 1983, are abrogated.*

In this example, the condition tends to transform the textual amendment into a material amendment which, on each occasion will be able to be represented with the schema of the derogation, of the constraint on effectiveness, etc, whilst in cases where the condition does not alter the textual nature of the amendment, reference will be made to the schemata presented here, with their appropriate adaptations.

5) *Relocation.* We found, even if sporadically, that there is the action of relocation[8] which will, probably, be inserted in our taxonomy as an autonomous category and will require us to write a new rule. In fact, this action, whilst not being foreseen by the drafting rules, seems to perform a specific legally and logically correct amending function, expressed with its own linguistic structure.

Finally, we found that some amendments were not recognised by the parser, not due to the inefficiency of the schemata we implemented, but for other factors (typographical mistakes in the text, linguistic errors, erroneous drafting with respect to the drafting rules, etc.).Therefore, we believe we have to make the rules formalised for recognising and extracting the amending provisions more flexible, also in all these cases where, for various reasons, the linguistic structure used in the text does not exactly correspond to the given model.

[8]For example: m) the letter h) of paragraph 5 of Article 20 is relocated as letter f), at the end of paragraph 1 of Article 17...

5 Applications and developments in the Project.

There are different types of applications where the legimatics methodologies described up until now can be used: 1) tagging and classifying the legislative text, 2) consolidating and co-ordinating legislative texts, 3) generating the legislative text, 4) evaluating and checking the legislative text.

About the evaluation and the check the legislative text, a project is now underway at ITTIG (Istituto di Teorie e Tecniche dell'Informazione Giuridica) for basing the checking functions, typical of Lexedit, on those of the Sophia 2.1 linguistic parser.

5.1 Lexedit XXI.

The project proposes a conceptual division of the functions to offer the user, into linguistic correction and structural validation. Here, we distinguish in the following between schemata linked to linguistic correction (primary) and structural validation (secondary).

5.1.1 Linguistic Correction

The central component of the architecture we propose resides in the client side and is a Verifier, which has the following responsibilities: communication with the graphic interface; communication with the Zoniser; communication with the Linguistic Corrector. The latter communication assumes that there is a communications protocol still to be determined. On the whole, the Verifier will transmit a paragraph in ASCI text and receive an XML document (or analogous structure) containing the linguistic errors which are discovered, together with their code and the position where they are identified.

The Zoniser in the schema above has the following responsibilities: to separate the text into paragraphs to be sent to the Linguistic Corrector; to identify errors which are not of a linguistic kind, such as wrong numbering, incorrect formatting, etc.

The Communications Server is a "light" component with the sole purpose of guaranteeing communications between the Verifier and the Linguistic Corrector, which is the central component of the server side sub-system[9]. This component will contain the business logic of the correction system and will manage the functions of the above three components, that is, Sophia 2.1, the Error Analyser and the Output Producer. Sophia 2.1 guarantees the application of one or more systems of rules aimed at identifying lack of conformity with the rules established by the legislative drafting technique. It will produce templates which identify any errors (Error Templates). The Error Analyser will analyse the sequence of Error Templates produced by Sophia 2.1 for the purpose of evaluating any contextual restrictions and for validating the actual errors, coupling them with an error code and enriching them with any data for their correction.

The objective of the Output Producer will be to produce a format that can easily be passed to the Verifier, which, in turn, will trigger a communications process with the user to help him/her correct any errors. Through the usual user interface, the user communicates his/her intention to correct the document. The order is passed to the Verifier which calls on the services of the Zoniser in order to identify the next paragraph to correct. It will be transformed into a textual format and passed to the Linguistic Corrector which, after analysing the text and the errors, will produce a document (in the abstract sense) containing the identified errors. This document will be passed back to the Verifier, which, through the user interface, will guide the user throughout the correction process.

[9]The term *Server* in this context is used to indicate the part of the system not directly accessible to components of the interface. It will therefore be a local server.

5.1.2 Structural Validation

The aim of the functions of Structural Validation is to guide the drafter/corrector towards a document that can be transformed automatically into an XML compliant with the DTD of NIR.

For the purposes of structural validation, the Zoniser has as its scope the identification of the structure of the document in terms of articles, paragraphs, etc... (where possible, that is, without resorting to linguistic analysis). For its part, X-Style will accept the document "reformatted" in this way in the rtf format (information about formatting derived from the Zoniser can be physically separated from the original document). X-Style's responsibilities will, therefore, be the following: to produce an XML document in output in keeping as far as possible with the DTD of NIR; to point out to the user (always through the Verifier) any anomalies or decisions that cannot be made automatically.

In order to implement the least intrusive process possible, we propose to use a Word template that guarantees convertability into XML, rather than forcing the user to work directly in XML. From the user's point of view, therefore, we plan to design the following functions:

1. The import phase (only once): the document will be translated from its original format into a Word document according to the NIR.dot template.
2. Validation: a document in NIR.dot will be evaluated in relation to its convertability into NIR.dtd.
3. Saving: a document validated according to NIR.dot will be able to be saved in XML in keeping with NIR.dtd without any further transformation.

References

[1] C. Biagioli, E. Francesconi, P.L. Spinosa, M. Taddei, The NIR Project. Standards and tools for legislative drafting and legal document Web publication, 2003 Proceedings of the International Conference of Artificial Intelligence and Law, Edinburgh, June 24, 2003.
[2] Marchetti, F. Megale, E. Seta, F. Vitali,Marcatura XML degli atti normativi italiani. I DTD di Norma in rete, in Informatica e diritto, 1, 2001, pp. 123-148.
[3] R. Pagano, Le direttive di tecnica legislativa in Europa, Camera dei deputati, Rome, 1997.
[4] S. Baroncelli, S. Faro, Tecnica legislativa e legislazione regionale: l'esperienza delle regioni Toscana, Emilia Romagna e Lombardia, in Iter Legis, 1998, pp. 173-213.
[5] G. Ferrari, Introduzione alla linguistica computazionale in http://apollo.vc.unipmn.it/~ling_gen/opening.htm;
[6] A. Bolioli, P. Mercatali, F. Romano, Formal Models for a Legislative Grammar. Explicit Text Amendment in M. A. Wimmer (edited by), Proceedings "Knowledge Management in Electronic Government (KMGov2004)", (Krems, Austria, May 17-19, 2004) Berlino, Springer.
[7] A. Bolioli, P. Mercatali, F. Romano, Legimatics Methodologies for the Implementation of a Legislative Grammar in atti del Workshop e-Government Modelling Norms and Concepts as Key Issues Edimburgo 24 June 2003.
[8] G. Sartor, Riferimenti normativi e dinamica dei nessi normativi, in Il procedimento normativo regionale, Cedam, Padua, 1996.
[9] M.C. De Lorenzo, Modelli di novelle, in Informatica e diritto, 1, 2002.
[10] P. Mercatali, Legimatica e nessi normativi, intervento al Seminario Nazionale di Studio "Formazione per le tecniche legislative", Turin, 17 and 18 June 1999, in Iter Legis, November – December 1999.
[11] P.L. Spinosa, I nomi uniformi dei provvedimenti giuridici adottati da "Norme in Rete", 2003 III Congresso mondiale di diritto e informatica, Havana 29 September-3 October 2003.
[12] M. Palmirani and R. Brighi Norma-System: A Legal Document System for Managing Consolidated Acts Database and Expert Systems *Applications* , 13th International Conference, DEXA 2002 Aix-en-Provence, France, September 2-6, 2002. Proceedings Springer-Verlag Heidelberg R. Cicchetti, A. Hameurlain, R. Traunmüller (Eds.).