Automatic semantics extraction in law documents

C.Biagioli, E.Francesconi
ITTIG - CNR
via Panciatichi 56/16, 50127, Firenze - Italy
http://www.ittig.cnr.it
{biagioli,francesconi}@ittig.cnr.it

A.Passerini
DSI - Univ. Firenze
via S. Marta 3, 50139 Firenze - Italy
http://www.dsi.unifi.it
passerini@dsi.unifi.it

S.Montemagni, C.Soria
ILC - CNR
via G.Moruzzi 1, 56100 Pisa - Italy
http://www.ilc.cnr.it
{montemagni,soria}@ilc.cnr.it

ABSTRACT

Normative texts can be viewed as composed by formal partitions (articles, paragraphs, etc.) or by semantic units containing fragments of a regulation (provisions). Provisions can be described according to a metadata scheme which consists of provision types and their arguments. This semantic annotation of a normative text can make the retrieval of norms easier. The detection and description of the provisions according to the established metadata scheme is an analytic intellectual activity aiming at classifying portions of a normative text into provision types and to extract their arguments. Automatic facilities supporting this intellectual activity are desirable. Particularly, in this paper, two modules able to qualify fragments of a normative text in terms of provision types and to extract their arguments are presented.

1. INTRODUCTION

The legal system usually suffers from scarce transparency which is caused by a non-systematic organization of the legal order. Law, in fact, is a normative and documentary unit of reference, hence the inability to obtain an analytical/systematic vision of a legal order itself, allowing to query a legal information system according to the content of each norm. This necessarily creates obstacles to the knowledge and upkeep of the legal order: from the uncertainty of the impact of new laws on the legal order in terms of coherency preservation, to the difficulties in norm accessing by both citizens and legal experts. For these reasons a more analytical unit of reference has been indentified in order to take a more organic view of the legal system [5, 7]. According to this point of view a normative text may be seen as a vehicle that contains and transports rules, or *provisions*, and the legal order as a set of rules rather than of laws. Recently, in Italy, the "Norme in Rete" (NIR) project ("Legislation on the Net") has adopted such a perspective within a context of strategies aiming at creating a unique access point of normative documents on the Web with search and retrieval services. A "provision-centric" view of legal order indeed has been considered of primary importance to define strategies and tools for the upkeep of legal systems and to provide facilities to access norms. Such strategies are essentially based on the identification of the provisions within a normative text and on providing them with an explicit semantic description in terms of *provision types* and their *arguments*, namely the actions and the entities, with their roles, which are regulated by the provisions. This activity can be carried out both manually and automatically. This paper explores automatic methodologies for helping the human activities of detecting the typologies of provisions contained in a normative document and extracting the related arguments. This paper is organized as follows: in Section 2 the main components of a model of provisions are introduced; Section 3 presents the standards established by the NIR project and the tools developed to make their adoption easier; among such tools in Section 4 a module able to automatically describe a normative document in terms of the contained provision types is described and tested, as well as in Section 5 a module able to automatically extract the arguments of the detected provisions is shown and evaluated. Finally, in Section 6 some conclusions are discussed.

ICAIL '05, June 6-11, 2005, Bologna, Italy.

2. A MODEL OF PROVISIONS

The entire body of the law, with its articles and paragraphs, may be seen as a set of provisions, intended as rules and carried by linguistic acts, and therefore propositions, whether simple or complex, endowed with meaning[1] [25]. Basically, a normative text can be viewed according to a structural or *formal profile*, and a semantic or *functional profile*. Following this perspective, fragments of a normative text are, at the same time, paragraphs and provisions, according to whether they are seen from a formal or functional viewpoint. The two points of view, both completely compatible with each other, can be alternated as required during the definition of the text. In particular the functional profile can also be considered as composed by two sub-profiles: the *regulative profile* and the *thematic profile*. The first one reflects the lawmaker directions, the second one the peculiarities of the regulated field. The regulative profile can be described in terms of *provisions*. A provision can assume different types as *definition*, *obligation*, *sanction*, *competence*, *amendaments*, etc. The thematic profile can be described by the so-called *arguments* of the provisions (for example the *ad-*

[1]Raz suggests among other things that a good criterion for individuation of the provisions should not depart too much or without well-founded reason from the ordinary concept of provision of law.

dressee of an obligation). *Types of provision* and related *arguments* define a model of provisions. They can be considered as a sort of metadata able to analitically describe the normative content of a normative text, hence the name of *analytical metadata*.
Using analytical metadata, a fragment of a normative text can be qualified according to a semantic point of view. For example, the following fragment of the Italian privacy law:

> *"A controller intending to process personal data falling within the scope of application of this Act shall have to notify the "Garante" ... "*

besides being considered as a part of the physical structure of a normative text (a *paragraph*), can also be viewed as a component of the logical structure of it (a *provision*). In particular, it can be qualified as a *provision* of type *obligation*, whose arguments are:

Addressee: "Controller";
Action: "Notification";
Third-party: "Garante".

The detection of provisions in a normative text consists, essentially, in an analytic effort in which all the possible distinctions among the elements, understood as rules, that go to make up the legislative texts, are made, and the nature and function of each one are singled out where possible.
The relevant issue is that of taking note of the types of rules in use and of defining their roles within a coherent functional vision of the legal system.
More in detail, it includes:

- Viewing the legal order as a rule system and the text as a set of rules;
- Clearly defining the rules functions in terms of provision types;
- As regards the internal structure of the rules, individuating the logically necessary components (argument types) of each type of provisions;
- As regards the relationships among the rules, analyzing the recurring and privileged relationships among types of rules.

The detection of the functional profile in a normative text can be carried out both during the drafting process, in order to more efficiently serve the needs of the legislator in his work of formulating and managing legal norms [5, 8], and at the end of the process as a pure documentalist activity.
Basically, such a function is similar to the qualification of a text using metadata according to a particular metadata scheme (for example the classification of a text according to the subject on the ground of a classification scheme). In our case the detection of the types of provisions and their arguments can be viewed as the process of qualification of a normative text using *analytical metadata*.
The use of analytical metadata permits different applications. Firstly, they allow to index and access normative documents from a semantic point of view. Legal persons, public administrations or associations handling personal data, for example, may query a legal information system where normative documents are indexed on the basis of the analytical metadata, searching for all the obligation regarding who is the controller of personal data, obtaining a selective answer.
Analytical metadata can also be used in the drafting phase giving the drafter the possibility to insert metadata annotion and also of drawing up new normative documents starting from their logical structure using an editor able to use metadata in the phase of building up a text. Moreover analytical metadata allows diagnosis on normative texts to analyse the coherency of the legal system.
Recently, within the NIR project, a standard to describe the functional profile of a normative text, as well as the formal one, have been defined for the Italian legal system.

3. THE NIR PROJECT: STANDARDS AND TOOLS

The "Norme in Rete" (NIR) project ("Legislation on the Net") has been proposed by the CNIPA [Italian National Center for Information Technology in the Public Administration] in conjunction with the Italian Ministry of Justice, with the aim of creating a unique access point on the Web with search and retrieval services of normative documents so as to eliminate the information historical fragmentation of the legal information systems, as well as to create a mechanism of stable cross-references able to guide users towards relevant sites of the public authorities participating in the project.
To achieve these purposes, the NIR project proposed to adopt a standardized description of normative documents able to guarantee interoperability among the sites and the information systems of the adhering authorities.
Particularly the project proposed the adoption of XML as a standard for representing normative documents [21] allowing the description of their formal and functional profiles (Section 2), and a uniform cross-referencing system, based on URN technique [29], providing documents with characteristics of interoperability and effective of use.
In order to make the adoption of such standards easier, a number of tools have been developed within the NIR project. The main one is *NIREditor* [8], an authoring tool which includes facilities, based on previous studies on legislative drafting [6], and modules aiming at managing new or legacy normative documents according to the established standards.
Hereinafter, two of these modules aiming at automatically detecting the functional profile in a normative text are described: they are the *Automatic Provision Classifier* (Section 4), able to classify law paragraphs as provision types (regulative profile (Section 2)), and the *Provision Argument Extractor* (Section 5), able to provide values to the arguments of the provisions (thematic profile (see Section 2)).

4. PROVISION AUTOMATIC CLASSIFIER

The detection of provisions in a normative text is basically a documentalist activity which is to be carried out by legal experts. However it can be particularly time consuming, especially for long and complex laws. An automatic module able to support the intellectual activity of classifying provisions is therefore desirable.
The *Provision Automatic Classifier* we have developed is a module able to automatically detect the type of provisions

contained in a normative text. In this work we have considered a paragraph as cointaining an entire provision, and also coincident with a provision, assumption, the second one, which is widely observed by the legislator. Assuming that a document has already been transformed in XML as concerns its physical structure (formal profile), this module classifies law paragraphs in terms of provision types, and it stores such information according to the NIR analytical metadata scheme. When combined with the *Provision Arguments Extractor*, it can detect the content of provision arguments, thus completing the detection of the normative text functional profile.

The Provision Automatic Classifier mainly consists of a text categorization algorithm which takes as input a law paragraph (hereinafter also called simply "document") d representing a provision, and outputs its predicted type (or "class") c choosing from a set of candidate classes C. In order to perform such an operation, it relies on a machine learning algorithm which has been trained on a set of training documents D with known class, and thus learned a model able to make predictions on new unseen documents. A wide range of machine learning approaches have been applied to automated text categorization, also in legal domain [16], and a vast literature on the subject exists (see [27] for a comprehensive review). Two correlated problems must be addressed in facing such a task: the choice of the document representation, that is how to turn the document into a format amenable for computation, and the choice of the particular learning algorithm to employ.

In Section 4.1 we outlined in detail the different types of document representation that we tried, while in Section 4.2 we described the *Multiclass Support Vector Machines* learning algorithm that was employed. Finally, Section 4.3 reports the experimental results of the method proposed.

4.1 Document Representation

A number of alternatives are possible in order to represent a document in a format which can be managed by an automatic classifier. Two main problems have to be faced: the choice of the meaningful textual units, representing the atomic *terms* of the document, and the level of structure to be mantained when considering the combination of such terms. Concerning the second problem, the most common approach, which we followed in our implementation, is that of ignoring the sequential order of the terms within a given document, and representing it simply as an unordered bag of terms. Concerning the first problem, the simplest possibility is that of representing words as terms, but more complex approaches can be conceived. A number of authors [11, 15] have tried using phrases as terms, but their experiments did not produce significantly better effectiveness. According to Lewis [20], a possible explanation for such a behaviour is that even if phrases have superior semantic qualities with respect to words, their statistical qualities are usually quite inferior. We thus limited ourselves to individual words in our document representation. Nevertheless, a number of preprocessing operations can be performed on pure words in order to increase their statistical qualities:

- *Stemming* can be applied to words in order to reduce them to their morphological root [2];
- Digit characters can be represented using a special character;
- Non alphanumeric characters can be represented using a special character.

Once basic terms have been defined, a vocabulary of terms $\mathcal{T}$ can be created from the set of training documents $\mathcal{D}$, containing all the terms which occur at least once in the set. A single document d will be represented as a vector of weights $w_1, \ldots, w_{|\mathcal{T}|}$, where the weight w_i represents the amount of information which the i^{th} term of the vocabulary carries out with respect to the semantics of d. We tried different types of weights, with increasing degree of complexity:

- A *binary* weight $\delta(w, d)$ indicating the presence/absence of the term within the document;
- A *term-frequency* weight $tf(w, d)$ indicating the number of times the term occurs within the document, which should be a measure of its representativeness of the document content;
- A *TFIDF* weight which indicates the degree of specificity of the term with respect to the document. Term Frequency Inverse Document Frequency [12] is computed as

  $$tfidf(w, d) = tf(w, d) * log(|D_w|^{-1})$$

  where $|D_w|$ is the fraction of training documents containing at least once the term w. The rationale behind this measure is that term frequency is balanced by *inverse document frequency*, which penalizes terms occurring in many different documents as being less discriminative.

Note that any learning algorithm to be run on the set of training examples D won't be able to compute statistics over terms not included in the vocabulary $\mathcal{T}$. Therefore, new test documents will still be represented as vectors of size $\mathcal{T}$, and any term not included in $\mathcal{T}$ will be ignored. Moreover, statistics computed for extremely rare terms will be far less reliable, as already pointed out for phrases with respect to words, thus possibly leading to *overfitting* phenomena. In order to address such a problem, *feature selection* techniques can be applied to reduce the number of terms to be considered, thus actually restricting the vocabulary to be employed (see e.g. [27, 32]). We tried two simple methods:

- An unsupervised *min frequency* threshold over the number of times a term has been found in the entire training set, aiming at eliminating terms with poor statistics.
- A supervised threshold over the Information Gain [24] of terms, which measures how much a term discriminates between documents belonging to different classes. The Information Gain of term w is computed as:

[2]We employed the snowball software, available at http://www.snowball.tartarus.org/italian/stemmer

$$ig(w) = H(D) - \frac{|D_w|}{|D|}H(D_w) - \frac{|D_{\bar{w}}|}{|D|}H(D_{\bar{w}})$$

where H is a function computing the entropy of a labelled set, D_w is the set of training documents containing the term w, and $D_{\bar{w}}$ is the set of training documents not containing it. Entropy in information theory measures the amount of bits necessary to encode the class of a generic element from a labelled set, and thus depends on the dispersion of labels within the set. Information Gain measures the decrease of entropy obtained by dividing the training set basing on the presence/absence of the term, thus preferring terms which produce subsets with more uniform labels.

4.2 Classification Algorithm

Binary classification is a typical machine learning task, and a number of different approaches have been developed so far. A main difference between classification algorithms is that of generative vs discriminative ones. The first type of algorithms learns a model for each possible label, and predicts the label of an example as the most likely given the example and the models. The second type directly learns the posterior probability of the label given the example, and is usually considered more appropriate for classification tasks (see e.g. [30]), even if different opinions arise for non asymptotic behaviours [22]. We focused on Support Vector Machines (SVM)[13, 10] as a state-of-art discriminative approach, provided with strong theoretical background and a number of successfull applications in various domains [26, 28]. The extension of binary classifiers to the multiclass case is straightforward for algorithms like decision trees [24], neural networks [9] or Bayesian classifiers [18], while SVM require more complex extensions. Generally speaking, SVM for multiclass classification have been developed as either combinations of binary classifiers, or by directly implementing multiclass versions of the SVM learning algorithm (see [17, 23] for reviews and comparisons). We choose the second methodology and employed a multiclass support vector machine (MSVM) as developed independently by Vapnik [30] and Crammer and Singer [14] (other implementations of MSVM [31, 19] differ in the cost function employed).
Assume a training set $D = \{(\mathbf{d}_i, c_i) \in \mathcal{D} \times [1, C]\}_{i=1}^{m}$ of examples belonging to one of C possible classes and represented as vectors in a vector space $\mathcal{D}$ of dimension $\mathcal{T}$. The decision function implemented by the MSVM algorithm is given by:

$$f(\mathbf{d}) = \text{argmax}_{c\in[1,C]}\langle \mathbf{w}_c, \mathbf{d}\rangle. \quad (1)$$

Here we have a vector of parameters $\mathbf{w}_c$ for each of the C possible classes, and its dot product with the example $\mathbf{d}$ measures the confidence that the example belong to class c. The decision function f assigns a new example $\mathbf{d}$ to the most confident class. Parameters $\mathbf{w}_c$ are learned from the training set D by solving a quadratic optimization problem which amounts at simoultaneously minimizing the complexity of the learned hypothesis and the number of errors committed on the training set (see [30, 14] for the details).

4.3 Experimental Results

A wide range of experiments was conducted over a dataset made of 582 paragraphs, here simply documents, distributed among 11 classes (Tab. 1), representing as many types of provisions. Paragraphs of the data set have been collected from italian legislative texts; each paragraph represents the minimal formal division containing a provision. They have been selected and labelled by legal experts according to the model of provisions discussed in Section 2.

Class labels	Provision types (classes)	Number of provisions (documents)
c_0	Repeal	70
c_1	Definition	10
c_2	Delegation	39
c_3	Delegification[3]	4
c_4	Prohibition	13
c_5	Reservation	18
c_6	Insertion	121
c_7	Obligation	59
c_8	Permission	15
c_9	Penalty	122
c_{10}	Substitution	111

Table 1: Provision types (classes) and number of provisions (documents) for each class in the experiments

In a preliminary phase, we found out that removing quoted sentences from documents before processing them led to better performances. In fact, they usually do not carry out semantic information regarding the type of provision in which they appear, and they would lead to poor statistics for the great variability of the text they can contain. After such a preprocessing step, we tried a number of combinations of the document representation and feature selection strategies described in Section 4.1, for the MSVM algorithms. We employed a *leave-one-out* (loo) procedure for measuring performances of the different strategies. For a dataset of n documents $D = \{d_1, \ldots, dn\}$, it consists of performing n runs of the learning algorithm, where for each run i the algorithm is trained on $D \setminus d_i$ and tested on the single left out document d_i. The loo accuracy is computed as the fraction of correct tests over the entire number of tests. Table 2 reports loo accuracy and train accuracy, which is computed as the average train accuracy over the loo runs, of the MSVM algorithm for the different document representation and feature selection strategies. The first three columns (apart from the index one) represent possible preprocessing operations. The fourth column indicates the term weighting scheme employed, binary (δ), term frequency (tf) and TFIDF ($tfidf$). The two following columns are for feature selection strategies: the unsupervised *min frequency* and the supervised *max infogain*, which actually indicates the number of terms to keep, after being ordered by Information Gain. Finally, the last two columns contain loo and train accuracies. While replacing digits or non alphanumeric characters slightly degrades performances, with a loo accuracy of 90.38% compared to 91.24%, the use of stemming actually helps clustering together terms with common semantics, bringing loo accuracy to 91.92%. The simpler binary weight scheme (loo acc. 91.92%) appears to work better then term frequency (loo acc. 88.14%) and tfidf (loo acc. 89.18%),

[3]Type of provision identifying matters that will be hereinafter regulated by administrative acts.

#	repl. digit	repl. alnum	use stem	weight scheme	min freq sel.	max IG sel.	loo acc (%)	train acc (%)
1	no	no	no	δ	no	no	91.24	100
2	yes	no	no	δ	no	no	90.38	100
3	yes	yes	no	δ	no	no	90.38	100
4	yes	yes	yes	δ	no	no	91.92	100
5	yes	yes	yes	δ	2	no	91.92	100
6	yes	yes	yes	tf	2	no	88.14	100
7	yes	yes	yes	$tfidf$	2	no	89.18	99.66
8	yes	yes	yes	δ	2	2000	91.92	100
9	yes	yes	yes	δ	2	1000	91.92	100
10	yes	yes	yes	δ	2	500	92.44	100
11	yes	yes	yes	δ	2	250	91.24	100
12	yes	yes	yes	δ	2	100	88.66	100
13	yes	yes	yes	δ	2	50	87.80	98.11

Table 2: Detailed results of MSVM algorithm for different document representation and feature selection strategies.

probably for the small size, in terms of number of words, of the provisions in our training set; this fact makes statistics on the number of occurences of a term less reliable. Only a slight improvement can be obtained by feature selection, bringing loo accuracy to 92.44% when choosing the 500 terms with maximum infogain. This confirms how SVM algorithms are able to effectively handle quite large feature spaces.

Classes	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_0	69	0	0	0	0	0	0	0	0	0	1
c_1	0	7	0	0	0	0	0	2	0	1	0
c_2	0	0	39	0	0	0	0	0	0	0	0
c_3	0	0	0	4	0	0	0	0	0	0	0
c_4	1	0	0	0	5	1	1	2	1	1	1
c_5	0	0	0	0	2	8	1	0	5	2	0
c_6	1	0	0	0	1	0	118	0	0	0	1
c_7	0	1	0	0	1	1	0	54	2	0	0
c_8	1	0	0	0	1	3	1	3	5	0	1
c_9	0	0	0	0	1	0	0	0	0	121	0
c_{10}	1	0	0	0	0	0	1	0	0	1	108

Table 3: Confusion matrix for the best MSVM classifier.

Classes	Precision	Recall	Entropy
c_0	0.945	0.986	0.120
c_1	0.875	0.700	0.157
c_2	1.000	1.000	0.000
c_3	1.000	1.000	0.000
c_4	0.455	0.385	0.642
c_5	0.615	0.444	0.430
c_6	0.967	0.975	0.079
c_7	0.885	0.915	0.200
c_8	0.385	0.333	0.509
c_9	0.960	0.992	0.092
c_{10}	0.964	0.973	0.085

Table 4: Precision, recall and entropy values for the best MSVM classifier.

Table 3 shows the confusion matrix for the best classifier, the MSVM indexed 10, reporting details of predictions for individual classes. Rows indicate true classes, while columns indicate predicted ones. Note that most errors are committed for classes with fewer documents, for which poorer statistics could be learned. Finally, in Table 4 we reported precision, recall and entropy values for each class.

5. PROVISION ARGUMENT EXTRACTOR

As mentioned above, paragraphs of normative documents can be analyzed not only according to the particular type of provision they express, but also with respect to the main legal entities involved by the law. Consistently, each provision type can be formally defined as a *frame* with a fixed number of (possibly optional) *slots*. The slot types required for the description of seven of the previously introduced classes of the provision model are illustrated in Table 5. The purpose of the *Provision Argument Extractor* is to select relevant text fragments corresponding to specific semantic roles[4] that are relevant for the different types of provisions[5]. In a nutshell, the Provision Argument Extractor is realized as a suite of Natural Language Processing tools for the automatic analysis of Italian texts (see [2]), specialized to cope with the specific stylistic conventions of the legal parlance. Although legal language is considerably more constrained than ordinary language, its specific syntactic and lexical structures still pose a considerable challenge for state-of-the-art Natural Language Processing tools. Nonetheless, if our goal is not a fully-fledged representation of their content, but only identification of specific information portions, legislative texts are relatively predictable and hence tractable through Natural Language Processing techniques.

Provision types	Arguments
Repeal	Rule, Position, *Novellando*
Prohibition	Action, Third-party
Insertion	Rule, Position, *Novella*
Obligation	Addressee, Action, Third-party
Permission	Addressee, Action, Third-party
Penalty	Addressee, Action, Object, Rule
Substitution	Rule, Position, *Novellando*, *Novella*

Table 5: A frame-based description of the different provision types.

A first prototype of the Provision Argument Extractor has just been brought to completion and its performance evaluated. The NLP technology used is relatively simple, but powerful, also thanks to the comparative predictability of normative texts. The Provision Argument Extractor module takes in input single law paragraphs in raw text, coupled with the categorization provided by the Automatic Provision Classifier, and outputs a semantic tagging of the text, where the semantic roles corresponding to different arguments are rendered as XML tags. An output example (translated into English for the reader's convenience) is given in Figure 1, where the input paragraph is classified as an *obligation* and portions of the text are identified as respectively denoting the *addressee* and the *action*. The approach to provision argument extraction from normative documents follows a two–stage strategy. In the first step, a general purpose parsing system, hand-tuned to handle some idiosyncrasies of Italian legislative texts, pre–processes each law paragraph to provide a shallow syntactic analysis. In the second step, the

[4]In order to make arguments useful for searching purposes it would be necessary not only to identify the text fragments but also to extract the relevant terms contained there and map them onto the corresponding entries in thesauri, dictionaries, or linguistic ontologies. The development of this project expects to provide the drafter with the possibility of inserting links to ontology concepts, with the twofold aim of a semantic indexing of normative documents on the one hand, and the population of domain-specific ontologies on the other.

[5]For more details about this module, the interested reader is referred to [4] and [3].

```
<obligation>
    <addressee>The Member State</addressee>
    shall
    <action>pay the advance within 30 calendar
            days of submission of the application
            for advance payment</action>.
</obligation>
```

Figure 1: Output example.

syntactically pre–processed text is fed into the semantic annotation component proper, making explicit the information content implicitly conveyed by the provisions.

5.1 Syntactic pre–processing

Syntactic pre–processing produces the data structures to which semantic annotation applies. At this stage, the input text is first tokenized and normalized for dates, abbreviations and multi–word expressions; the normalized text is then morphologically analyzed and lemmatized, using an Italian lexicon specialized for the analysis of legal language; finally, the text is POS-tagged and shallow parsed into non–recursive constituents called "chunks". A chunked sentence, however, does not give information about the nature and scope of inter–chunk dependencies. These dependencies, whenever relevant for semantic annotation, are identified at the ensuing processing stage (see section 5.2 below).
Although full text parsing may be suggested as an obvious candidate for adequate content processing, we contend that shallow syntactic parsing provides a useful intermediate representation for content analysis. First, at this stage information about low level textual features (e.g. punctuation) is still available and profitably usable, whereas it is typically lost at further stages of analysis. In this connection, it should be appreciated that correct analysis of modifications crucially depends on punctuation marks, and in particular on quotes and colons, which are used to identify the text of the amendment (*novella*) and the amending text (*novellando*). Secondly, chunked sentences naturally lend themselves to incrementally being used as the starting point for partial functional analysis, whereby the range of dependency relations that are instrumental for semantic annotation is detected. In particular, dependency information is heavily used for the mark–up of both modifications and obligations, which requires knowledge of the underlying syntactic structure. Finally, a third practical reason is that chunking yields a local level of syntactic annotation. As a result, it does not "balk" at domain–specific constructions that violate general grammar patterns; rather, parsing is carried on to detect the immediately following allowed structure, while ill–formed chunks are left behind, unspecified for their category.

5.2 Semantic annotation

As mentioned above, the implementation of the Provision Argument Extractor module is closely inspired by mainstream techniques of Information Extraction. In particular, semantic annotation consists in the identification of all arguments relevant to a specific provision type. A provision type can then be regarded as a frame, which in turn acts as an *extraction template* whose slots are filled with the textual material matching the corresponding conceptual roles.
The semantic annotation component takes in input a chunked representation of each law paragraph and identifies semantically relevant structures by applying domain–dependent, finite state techniques locating relevant patterns of chunks. Semantic mark–up is performed through a two–step process:

1. Each paragraph is assigned to a frame (corresponding to the legislative provision expressed in the text);
2. Slots of the frame identified at step (1) are turned into an extraction template and instantiated through the structural components (i.e. sentences, clauses, phrases) of the law paragraph.

The current version of the semantic annotation component is a specialized version of the ILC finite–state compiler of grammars for dependency syntactic analysis (see [1]). The specific version of the grammar compiler uses a specialized grammar including (i) a core group of syntactic rules for the identification of basic syntactic dependencies (e.g. subject and object), and (ii) a battery of specialized rules for the semantic annotation of the text.
All rules in the grammar are written according to the following template:

> <chunk-based regular expression> WITH <battery of tests> => <actions>

The extraction of provision arguments from law paragraphs is based on structural patterns that are combined with lexical conditions and other tests aimed at detecting low–level textual features (such as punctuation) as well as specific syntactic structures (e.g. the specification of a given dependency relation). Structural patterns are expressed in terms of regular expressions over sequences of chunks, whereas all other conditions (*e.g.* lexical, syntactic, etc.) are checked through a battery of tests. The action type ranges from the identification of basic dependency relations (in the case of syntactic rules) to the semantic mark–up of the text (in the case of semantic annotation rules).
Reliable identification of dependency relations is also important for assigning structural elements to semantic roles, since the latter tend to be associated with specific syntactic functions (e.g. subject, object). To give the reader but one example, the addressee of an obligation typically corresponds to the syntactic subject of the sentence, while the action (s)he is obliged to carry out is usually expressed as an infinitival clause, as in the example reported below:

> [[Il comitato misto]subj]addressee] è tenuto [[a raccomandare modifiche degli allegati secondo le modalità previste dal presente accordo]i_clause]action]
> [[*The Joint Committee*]subj]addressee] *shall* [[*be responsible for recommending amendments to the Annex as foreseen in this Agreement*]i_clause]action].

Note, however, that this holds only when the verbal head of the infinitival clause is used in the active voice. By contrast,

Provision Class	Success	Partial Success	Failure
Repeal	95.71%	2.86%	1.43%
Prohibition	73.33%	26.67%	–
Insertion	97.48%	1.68%	0.84%
Obligation	88.89%	11.11%	–
Permission	66.67%	20%	13.33%
Penalty	47.93%	45.45%	6.61%
Substitution	96.40%	3.60%	–
Tot.	**82.09%**	**15.35%**	**2.56%**

Table 6: Provision Argument Extractor results

the syntactic subject can express the action if the verb is used in the passive voice and is governed by specific lexical heads.

5.3 Experimental results

The Provision Argument Extractor module was evaluated on a sub–set of the dataset used for the evaluation of the Automatic Classifier. The evaluation set for the Argument Extractor consisted of 473 law paragraphs, covering seven provision classes of those defined by the NIR standards. Since the performance of the Argument Extractor is to be compared against human annotation, the choice of a reduced data set is justified by the need for restricting the costly process of deep, expert annotation to the most representative examples.

Table 6 illustrates the performance of the system. Knowing the right provision type of each example, the aim of the evaluation here was to assess the system's reliability in identifying, for each provision type or frame, all the semantic roles or arguments that are relevant for that frame and are instantiated in the text. Three possible cases are distinguished: a) the module correctly identifies all and only the relevant semantic roles instantiated in the provision text ("Success"); b) the module identifies only a subset of the relevant semantic roles ("Partial Success"); c) the module utterly fails and no role is detected.

6. CONCLUSIONS

The legal system usually suffers from scarce transparency mainly caused by a non-systematic organization of the legal order. Law is usually referred in terms of a normative and documentary unit, hence the difficulties in norm accessing by both citizens and legal experts. For these reasons it can be useful to identify a more analytical unit of reference, the *provision*, in order to take a more organic view of the legal system. Provisions can be described according to a model which consists of provision types and their arguments. Even if the detection of the provisions within a normative text is essentially an intellectual activity aiming at classifying portions of normative texts into provision types and to extract their arguments, automatic facilities supporting the user are desirable. In this paper two modules able to classify fragments of normative texts into provision types and to extract their arguments have been presented. They are based on multiclass Support Vector Machine classification techniques and on Natural Language Processing techniques respectively. The test of these two approaches produced promising results.

7. ACKNOWLEDGMENTS

A special thank goes to Stefano Pietropaoli Ph.D. student at the Law Faculty of the University of Pisa for his activity in collecting and analysing provision examples to train the two modules.

8. REFERENCES

[1] R. Bartolini, A. Lenci, S. Montemagni, and V. Pirrelli. Grammar and lexicon in the robust parsing of italian: Towards a non-naïve interplay. In *Proceedings of Coling 2002 Workshop on Grammar Engineering and Evaluation*, 2002.

[2] R. Bartolini, A. Lenci, S. Montemagni, and V. Pirrelli. The lexicon-grammar balance in robust parsing of italian. In *Proceedings of 3rd International Conference on Language Resources and Evaluation*, 2002.

[3] R. Bartolini, A. Lenci, S. Montemagni, V. Pirrelli, and C. Soria. Automatic classification and analysis of provisions in italian legal texts: a case study. In *Proceedings of the Second International Workshop on Regulatory Ontologies*, 2004.

[4] R. Bartolini, A. Lenci, S. Montemagni, and C. Soria. Semantic mark-up of legal texts through nlp-based metadata-oriented techniques. In *Proceedings of 4rd International Conference on Language Resources and Evaluation*, 2004.

[5] C. Biagioli. Definitional elements of a language for representation of statutory. *Rechtstheorie*, 11, 1991.

[6] C. Biagioli. Law making environment. In *Proceedings of Workshop on Legal Knowledge and Legal Reasoning Systems, Tokyo*, 1992.

[7] C. Biagioli. Towards a legal rules functional micro-ontology. In *Proceedings of workshop LEGONT '97*, 1997.

[8] C. Biagioli, E. Francesconi, P. Spinosa, and M. Taddei. The nir project: Standards and tools for legislative drafting and legal document web publication. In *Proceedings of ICAIL Workshop on e-Government: Modelling Norms and Concepts as Key Issues*, pages 69–78, 2003.

[9] C. Bishop. *Neural networks for pattern recognition*. Oxford University Press, Oxford, 1995.

[10] C. Burges. A tutorial on support vector machines for pattern recognition. In *Data Mining and Knowledge Discovery*. Kluwer Academic Publishers, Boston, 1998. (Volume 2).

[11] S. W. C. Apté, F.J. Damerau. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994.

[12] G. S. C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[13] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.

[14] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal on Machine Learning Research*, 2:265–292, 2002.

[15] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, New York, NY, USA, 1998. ACM Press.

[16] A. R. E. Schweighofer and M. Dittenbach. Automatic text representation, classification and labeling in european law. In *Proceedings of the Eighth International Conference on Artificial Intelligence and Law*, pages 78–87, 2001.

[17] C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[18] F. Jensen. *Introduction to Bayesian Networks*. Springer-Verlag, 1996.

[19] Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines. Technical Report 1043, Dept. of Statistics, University of Wisconsin, 2001.

[20] D. Lewis. Automating the construction of internet portals with machine learning. In *Proceedings of ACM International Conference on Research and Development in Information Retrieval*, pages 37–50, 1992.

[21] F. Megale and F. Vitali. I dtd dei documenti di norme in rete. *Informatica e Diritto*, 1:167–231, 201.

[22] A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[23] A. Passerini. *Kernel Methods, Multiclass Classification and Applications to Computational Molecular Biology*. Ph.d thesis, Università di Firenze, Italy, 2004.

[24] J. Quinlan. Inductive learning of decision trees. *Machine Learning*, 1:81–106, 1986.

[25] J. Raz. *The Concept of a Legal System*. Oxford University Press, 1980.

[26] B. Schölkopf and A. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

[27] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[28] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[29] P. Spinosa. Identification of legal documents through urns (uniform resource names). In *Proceedings of the EuroWeb 2001, The Web in Public Administration*, 1997.

[30] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[31] J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science, 1998.

[32] Y. Yang and J. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420. Morgan Kaufmann Publishers Inc., 1997.