Artificial Intelligence and Law
DOI 10.1007/s10506-007-9038-0

Automatic Classification of Provisions in Legislative Texts

E. FRANCESCONI[1] and A. PASSERINI[2]

[1]*ITTIG - CNR, Istituto di Teoria e Tecniche dell'Informazione Giuridica - Consiglio Nazionale delle Ricerche, Via Barucci 20, Florence, Italy*
E-mail: francesconi@ittig.cnr.it
[2]*DSI - Dipartimento di Sistemi e Informatica, Università di Firenze, Via S. Marta, 3, 50139, Florence, Italy*
E-mail: passerini@dsi.unifi.it

Abstract. Legislation usually lacks a systematic organization which makes the management and the access to norms a hard problem to face. A more analytic semantic unit of reference (*provision*) for legislative texts was identified. A *model of provisions* (provisions types and their arguments) allows to describe the semantics of rules in legislative texts. It can be used to develop advanced semantic-based applications and services on legislation. In this paper an automatic bottom-up strategy to qualify existing legislative texts in terms of provision types is described.

Key words: model of provisions, Naïve Bayes classifier, SVM classifier, text categorization

1. Introduction

The lack of knowledge and control of the legal order, which the scarce transparency of the legal system depends on, represents a crucial problem for both the legislator and the citizens. Problems of different nature arised in dealing with a non-systematic organization of legal order, from the uncertainty of the impact of new laws in terms of coherency preservation, to the difficulties in norm accessing by both citizens and legal experts.

In the '90 Biagioli (1991, 1997) identified a possible reason of these problems in the fact that while a law is a normative and documentary unit of reference, users and legal experts usually manage, access and refer to the legal order in terms of the contained norms.

The inability to obtain an analytical/systematic vision of a legal order necessarily creates obstacles to its knowledge and upkeep. Therefore a more analytical unit of reference was identified in order to have a more organic view of the legal system. According to this point of view a legislative 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. This perspective, inspired by analytic legal philosophy, permits to perceive the rules as the true bricks in the legal system, and the laws as purely temporary events.

Recently, in Italy, the "Norme in Rete" (NIR) project ("Legislation on the Net") has been launched with the aim of defining standards for the Italian legislation, allowing the creation of a unique access point for legal documents in a distributed environment with search and retrieval facilities, as well as a mechanism of stable cross-references able to guide users towards relevant sites of public authorities adhering to the project. To achieve these purposes the NIR project proposed the adoption of XML as a standard for representing legislative texts (Megale and Vitali 2001) and URN technique as standard to univocally identify legal measures (Spinosa 2001). In order to pave the way to advanced applications and services on legislation (from semantic search and retrieval facilities, to consolidation) a rich set of metadata has been defined as well: besides general information on the acts using general metadata, also the semantics of rules in legislative texts can be described using a *provision* model: a "provision-centric" view of legal order in fact has been considered of primary importance to define strategies and tools for the upkeep of legal systems and to provide facilities to access norms.

This paper is particularly addressed in exploring automatic methodologies able to help the human activities of classifying legislative texts according to the model of provisions. This paper is organized as follows: in Section 2 the main components of the model of provisions are introduced; in Section 3 an overview of the standards for Italian legislation established by the NIR project, as well as the tools developed to make the adoption of such standards easier, are presented; among such tools, in Section 4 a module able to automatically classify legislative texts in terms of provisions is presented. The approach adopted for representing documents in a way amenable for computation and the methodologies used to implement the automatic provision classifier are shown in Sections 5 and 6 respectively. Finally, in Section 7 a comparison of the experimental results on a set of documents is reported and in Section 8 some conclusions are discussed.

2. The Model of Provisions

The entire body of laws and regulations, with their articles and paragraphs, may be seen as a set of *provisions*, carried by speech acts, namely sentences, whether simple or complex, endowed with meaning (Raz 1977). Basically, a legislative text can be viewed according to a structural or *formal profile*, and a semantic or *functional profile* (Biagioli 1997). Following this perspective, fragments of a legislative text are, at the same time, paragraphs and

provisions, according to whether they are seen from a formal or functional view-point.

In particular the functional profile of a legislative text can be more analytically analyzed till distinguishing in it two sub-profiles representing as many specific semantic roles: 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 *provision types*. A provision can assume different types as *definition*, *obligation*, *sanction*, *competence*, *amendments*, etc. The thematic profile can be described by the so-called *arguments* of the provisions (for example the *addressee* of an obligation).

Provision types and related *arguments* represent a semantic model of legislative texts which has been called *model of provisions* (Biagioli 1997). They can be considered as a sort of metadata scheme able to describe analytically the content of a legislative text, hence also the name of *analytical metadata*.

Using analytical metadata, a fragment of a legislative text can be qualified, from a semantic point of view, according to the model of provisions. 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'' thereof, …* ''

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

Addressee:	''Controller'';
Action:	''Notification'';
Counter-party:	''Garante''.

The use of the provision model to describe the semantics of legislative texts gives the possibility of developing different applications. A corpus of laws and regulations entirely qualified according to the model of provisions allows to develop advanced search and retrieval services for legislative documents (Biagioli and Turchi 2005). Moreover it allows to perform analyses concerning the coherency of the legal system. The model of provision can be used in top-down or bottom-up strategies. When used in a top-down strategy it provides facilities to draft new bills from a semantic point of view (Biagioli and Francesconi 2005), thus contributing to enhance the quality of legislative texts. In a bottom-up strategy it can be used to describe existing legislative texts in a pure documentalistic activity.

Particularly the bottom-up strategy can be carried out manually or automatically. The manual bottom-up detection of provisions in an existing text consists, essentially, in an analytic effort in which all the possible distinctions among the elements that go to make up a legislative text are identified, singling out the nature and function of each. Basically, on the basis of the model of provisions, this activity consists in classifying portions of a legislative text according to a *provision type* and in defining the roles of text fragments in terms of *provision arguments*, within a coherent functional vision of the legal system. The automatic (or semi-automatic) bottom-up detection of provisions consists in using tools able to detect and classifying provisions, as well as to extract their arguments.

Basically, the bottom-up strategy 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). In our case the detection of the types of provisions and their arguments can be viewed as the process of qualification of a legislative text using *analytical metadata*.

Recently, within the NIR project, the model of provisions has been considered as standard to describe the functional profile of Italian legislative texts. In Section 3 an overview of the NIR project is shown as regards the established standards and the tools developed to make their adoption easier.

3. The NIR project: standards and tools

The ''Norme in Rete'' (NIR) project (''Legislation on the Net'') has been proposed by 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 legislation 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. Similar initiatives are the MetaLex project in the Netherlands (Boer et al. 2002), the LexDania project in Denmark, CHLexML in Switzerland, the e-Recht project in Austria.

3.1. THE NIR STANDARDS

To achieve its purposes, the NIR project proposed to adopt a standardized description of legislative documents able to guarantee interoperability among sites and information systems of the adhering authorities. Two national work groups produced two main official standards:

1. a standard for legislative document identification defined in accordance with the uniform name (URN) technique: an unambiguous identifier, that allows the references to be expressed in a stable way, irrespective of their physical location (Spinosa 2001);
2. a standard for legislative document representation, using XML techniques. In particular three DTDs (NIR-DTDs) of increasing degree of complexity in text hierarchy description (Megale and Vitali 2001) have been defined. NIR-DTDs are able to describe:
   - the structure of legislative texts (*formal profile*), establishing constraints in the hierarchy of the formal elements (basically headings, closing formula, collections of articles, etc.);
   - the semantics of legislative texts, describing general information on a whole text using general metadata (subject matter, publication date, etc.), and fragments of the contained regulation (*functional profile*) described by the *model of provisions* (type of provisions and their arguments), in terms of XML-NIR analytical metadata scheme.

In order to make the adoption of such standards easier, ITTIG-CNR developed a number of tools implementing such standards. The main one is *NIREditor*, a legislative drafting environment based on previous studies on legislative drafting (Biagioli 1992), which includes facilities aiming at managing new or legacy legislative texts according to the established standards. Within the NIR project other tools have been developed by different institutes supporting the implementation of the NIR standards (as the *Norma-Editor* developed by CIRSFID (Palmirani 2005)).

3.2. THE NIREDITOR

The *NIREditor* is a legislative drafting environment, operating within the URN and DTD NIR framework, supporting specific legislative technique functions (Biagioli et al., 2005b). It is designed to assist the production of new texts, as well as to process legacy contents.

A particular attention has been addressed to design automatisms for legacy content handling, since they represent key-factors for promoting the adoption of the standards. They include four main modules:

1. the *Cross-Reference Parser*, designed to detect cross-references and to construct the related URNs;
2. the *Structure Parser*, designed to automate the XML-NIR conversion of legacy contents;
3. the *Provision Automatic Classifier*, which automatically classifies paragraphs into provision types according to the model of provisions;

4. the *Provision Arguments Extractor*, which automatically extracts the arguments of the provisions.

The first three modules (Biagioli et al., 2005b) have been developed by ITTIG—CNR in collaboration with Computer Science Department (DSI) of the University of Florence, while the Provision Arguments Extractor has been developed by the Institute of Computational Linguistic of the Italian National Research Council (ILC—CNR) (Bartolini et al., 2004).

In particular the last two modules are able to deal with the automatic, or semi-automatic, bottom-up detection of semantics in legislative texts, implementing the model of provisions in terms of XML-NIR analytical metadata scheme discussed in Section 3.1. Hereinafter our implementation of the *Provision Automatic Classifier* is presented. For the implementation of the other modules see (Biagioli et al., 2005b) and (Biagioli et al., 2005a).

4. Provision automatic classifier

The bottom-up detection of provisions in a legislative text is an activity which usually requires legal expertise. Moreover it can be particularly time consuming, especially for long and complex texts. An automatic system able to support the intellectual activity of classifying provisions is therefore desirable.

The *Provision Automatic Classifier* we developed is a module able to automatically detect the type of provisions contained in legislative texts. The basic assumption we have considered, widely observed by the legislator, is that a ''paragraph'', the basic component of the formal profile, usually contains a ''provision'', basic component of the functional profile. Having a legislative document in XML-NIR format, this module classifies paragraphs in terms of provision types, and it inserts such information in the current document according to the XML-NIR analytical metadata scheme. This module, combined with the *Provision Arguments Extractor*, contributes to describe the semantics (functional profile) of a legislative text. The fragment (article 7, paragraph 1 of the Italian privacy law) reported in Section 2 will be qualified according to the XML-NIR analytical metadata scheme (localized in English) as follows:

```
<dsp:obligation>
      <dsp:pos xlink:type="simple" xlink:href="#art7-com1"/>
      <dsp:addressee> Controller </dsp:addressee>
      <dsp:action> Notification </dsp:action>
      <dsp:counter-party> Garante </dsp:counter-party>
</dsp:obligation>
```

The Provision Automatic Classifier mainly consists of a text categorization algorithm which takes as input a text paragraph (hereinafter 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 machine learning algorithms which have 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, and a vast literature on the subject exists (see (Sebastiani 2002) 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 5 we outline in detail the different types of document representation we tried, while in Section 6 we describe two learning algorithms, *Naïve Bayes* and *Multiclass Support Vector Machines*, we have used. Finally, Section 7 reports a detailed experimental comparison of the different methods and representations proposed.

5. 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 maintained 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 (Apté 1994; Dumais et al. 1998) have tried using phrases as terms, but their experiments did not produce significantly better effectiveness. According to Lewis (1992), 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.[1]
- 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 ith 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 (Buckley 1988) 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. Sebastiani 2002; Yang and Pedersen 1997). 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 (Quinlan 1986) of terms, which measures how much a term discriminates between documents belonging to different classes. The Information Gain of term w is computed as:

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

where H is a function computing the entropy of a labelled set ($H(D) = \sum_{i=1}^{|C|} -p_i \, log_2(p_i)$, being p_i the portion of D belonging to class i), D_w

is the set of training documents containing the term w, and $D_{\bar{w}}$ is the set of training documents not containing w. 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. Basically it measures the discriminative power of a term, with respect to different classes; in other words it measures the effectiveness of an attribute in classifying the training data. In fact, given a term w and a set of data, labelled as positive (S_+) and negative (S_-) examples, the optimal case for the information gain value of w is represented by the situation in which all the documents containing w belong to a single specific class, say S_+, and all the documents which do not contain w belong to S_-, (in our case the entropies of the two sets of documents $H(D_w)$ and $H(D_{\bar{w}})$ are 0 and the information gain $ig(w)$ is maximum ($ig(w) = H(D)$)). This method basically allows to select terms with the highest discriminatory power among a set of classes.

6. Classification algorithms

Binary classification is a typical machine learning task, and a number of different approaches have been developed so far. Its extension to the multiclass case is straightforward for algorithms like decision trees (Quinlan 1986), neural networks (Bishop 1985) or Bayesian classifiers (Jensen 1996), while algorithms like Support Vector Machines (SVMs) (Cortes and Vapnik 1995) require more complex extensions (see Passerini 2004 for a review). 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. Vapnik 1998), even if different opinions arise for non asymptotic behaviours (Ng and Jordan 2002). We employed a simple generative approach, called *Naïve Bayes*, which proved quite effective for text categorization (Joachims, 1997), and a multiclass extension of the SVMs as a state-of-art discriminative approach.

6.1. NAÏVE BAYES CLASSIFIER

A Bayesian classifier is a probabilistic classifier which outputs the class with maximum a posteriori probability, computed using the Bayes theorem:

$$P(c_i|d) = \frac{P(c_i)P(d|c_i)}{P(d)}$$

Given the vectorial representation of the document described in the previous section, the Naïve assumption states that the probability of individual terms within a document is independent of each other given the class. The class with maximum a posteriori probability is thus computed as:

$$c^* = \arg max_{c_i \in C} P(c_i) \prod_{k=0}^{|d|} P(w_k|c_i) \tag{1}$$

where the product runs over the terms contained in both d and the vocabulary $\mathcal{T}$. Terms can be considered either once only or once for each occurrence in the document, thus implementing the first and second term weighting schemes described in the previous section.

Let $D_i \subseteq D$ be the subset of training documents belonging to class c_i. Probabilities are computed during the training phase in the following way:

– The prior probability of class c_i is computed as the fraction of training documents D_i belonging to it:

$$P(c_i) = \frac{|D_i|}{|D|}$$

– The probability of an individual term w_k given a class c_i is computed as:

$$P(w_k|c_i) = \frac{n_k + 1}{n + |\mathcal{T}|}$$

where n is the total number of term occurrences in documents D_i, and n_k is the number of times term w_k occur in documents D_i.

Note that the second definition includes the so called *laplace smoother*, which is a uniform prior distribution of terms over the vocabulary $\mathcal{T}$.

6.2. MULTICLASS SUPPORT VECTOR MACHINE CLASSIFIER

Kernel Methods (Schölkopf and Smola 2002; Shawe-Taylor and Cristianini 2004) are an established methodology for statistical learning, provided with strong theoretical background and a number of successful applications in various domains. The first implementation of such a methodology was that of SVMs (Cortes and Vapnik 1995; Burges 1998) for binary classification tasks. Extensions to the multiclass classification case have been developed as either combinations of binary classifiers, or by directly implementing a

multiclass version of the SVM learning algorithm (see (Hsu and Lin 2002; Passerini 2004) for reviews and comparisons). We choose the second methodology and employed a multiclass support vector machine (MSVM) as developed independently by Vapnik (1998) and Crammer and Singer (Crammer and Singer 2002) (other implementations of MSVM (Weston and Watkins 1998; Lee et al. 2001) differ in the cost function employed). In the following we will briefly recall the MSVM learning algorithm, limiting ourselves to the unbiased linear case which is a common choice for text categorization tasks.

Assume a training set $D = \{(d_i, c_i) \in \mathcal{D} \times [1, C]\}_{i=1}^{m}$ of examples belonging to one of C possible classes.[2] The decision function implemented by the MSVM algorithm is given by:

$$f(\mathbf{d}) = \text{argmax}_{c \in [1,C]} \langle \mathbf{w}_c, \mathbf{d} \rangle, \tag{2}$$

where $\langle \mathbf{w}_c, \mathbf{d} \rangle$ denotes the dot product between vectors $\mathbf{w}_c$ and $\mathbf{d}$ of size $\mathcal{T}$. Here we have a vector of parameters $\mathbf{w}_c$ for each of the C possible classes, which determines a separating hyperplane in the vector space $\mathcal{D}$ given by $\mathbf{d} \in \mathcal{D} : \langle \mathbf{w}_c, \mathbf{d} \rangle = 0$. Examples $\mathbf{d}$ for which $\langle \mathbf{w}_c, \mathbf{d} \rangle > 0$ are likely to belong to class c, while those for which $\langle \mathbf{w}_c, \mathbf{d} \rangle < 0$ have a negative evidence with respect to the class. The decision function f assigns a new example $\mathbf{d}$ to the most confident class.

The optimization problem solved by the MSVM algorithm in order to learn the parameters $\mathbf{w}$ is written as:

$$\min_{\mathbf{w} \in \mathcal{D}^C, \xi \in R^m} \quad \frac{1}{2} \sum_{c=1}^{C} ||\mathbf{w}_c||^2 + \frac{1}{\lambda} \sum_{i=1}^{m} \xi_i \tag{3}$$

subject to

$$\langle \mathbf{w}_{c_i}, \mathbf{d}_i \rangle - \langle \mathbf{w}_p, \mathbf{d}_i \rangle \geq 1 - \xi_i, \quad i \in [1, m], p \in [1, C] \backslash \{c_i\} \tag{4}$$

$$\xi_i \geq 0, \quad i \in [1, m]. \tag{5}$$

The function to be minimized (3) consists of two complementary terms: the first one aims at minimizing the square norm of the vectors of parameters $\mathbf{w}$, thus preferring simpler hypotheses to more complex ones. The second term accounts for errors committed on the training set, and aims at fitting training data. The regularization hyper-parameter λ trades off between these two complementary requirements. Constraints (4) state that the confidence of the correct assignment c_i for a given example $\mathbf{d}_i$ should be greater than any other assignment $p \neq c_i$ by at least one, suffering a linear penalty ξ_i for the greatest violation if any. The value one for the confidence, also known as

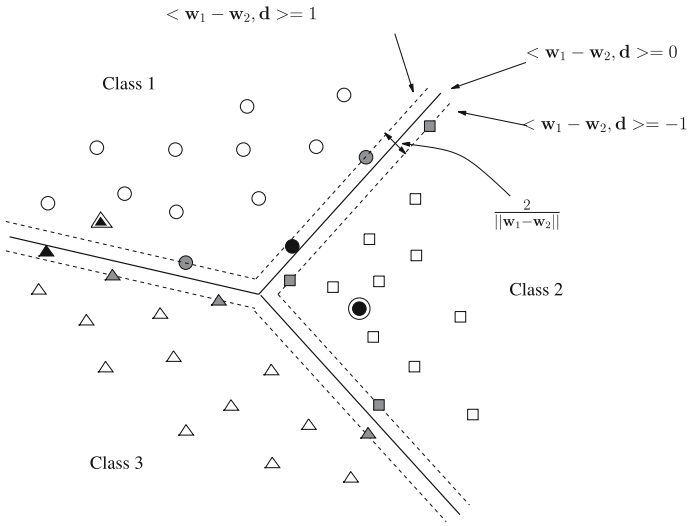

Figure 1. Multiclass classification problem solved by multiclass support vector machines (MSVMs). Solid lines represent separating hyperplanes, while dotted lines are hyperplanes with confidence margin equal to one. The multiclass margin is the minimal distance between two dotted lines. Dark points are support vectors. Black points are also constraints violations and extra borders indicate violations which are also training errors.

confidence margin, is a conventional choice necessary in order that the solution to the optimization problem is unique.

Figure 1 shows an example of learned hypothesis, where we can distinguish violations and *support vectors*, which are the subset of training examples responsible for the learned decision function. It also shows how the algorithm actually learns large margin hypotheses, that is separating hyperplanes with a large *multiclass margin*, indicated as the minimal distance between confidence one class hyperplanes.

7. Experimental results

A wide range of experiments was conducted over a dataset made of 582 provisions distributed among 11 classes (Table I), representing as many types of provisions.

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

Table I. Classes (provision types) and number of documents for each class in the experiments

Class labels	Classes of the data set	Number of documents
$c0$	Repeal	70
$c1$	Definition	10
$c2$	Delegation	39
$c3$	Delegification	4
$c4$	Duty	13
$c5$	Exception	18
$c6$	Inserting	121
$c7$	Prohibition	59
$c8$	Permission	15
$c9$	Penalty	122
$c10$	Substitution	111

feature selection strategies described in Section 5, for both the Naïve Bayes and MSVM algorithms. We employed a *leave-one-out* (loo) procedure for measuring performances of the different strategies and algorithms. For a dataset of n documents $D = \{d_1, \ldots, d_n\}$, 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 II reports loo accuracy and train accuracy, which is computed as the average train accuracy over the loo runs, of the Naïve Bayes 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, which for the Naïve Bayes are binary (δ) and term frequency (*tf*). 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 does not improve performances, the use of stemming actually helps clustering together terms with common semantics. The simpler binary weight scheme appears to work better then term frequency, while significant improvements can be obtained by performing feature selection with Information Gain.

Table III contains the same set of experiments for the MSVM algorithms, plus the case where TFIDF is used as a term weighting scheme (*tfidf*). Note that loo accuracy is consistently better for all kinds of strategy, thus confirming the effectiveness of SVM algorithms with respect to simple generative models for discriminative tasks. The binary weight scheme still

Table II. Detailed results of Naïve Bayes algorithm for different document representation and feature selection strategies

#	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	79.38	93.64
2	Yes	No	No	δ	No	No	79.21	91.92
3	Yes	Yes	No	δ	No	No	79.38	91.92
4	Yes	Yes	Yes	δ	No	No	83.33	92.10
5	Yes	Yes	Yes	δ	2	No	84.71	94.50
6	Yes	Yes	Yes	*tf*	2	No	82.47	92.44
7	Yes	Yes	Yes	δ	2	2000	84.71	94.50
8	Yes	Yes	Yes	δ	2	1000	85.57	95.88
9	Yes	Yes	Yes	δ	2	500	88.66	95.53
10	Yes	Yes	Yes	δ	2	250	88.14	94.33
11	Yes	Yes	Yes	δ	2	100	86.25	91.24
12	Yes	Yes	Yes	δ	2	50	85.22	87.80

appears to be the best one, 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 occurrences of a term less reliable. Only a slight improvement can be obtained in this case by feature selection, thus confirming how SVM algorithms are able to effectively handle quite large feature spaces.

Table III. Detailed results of multiclass support vector machine (MSVM) algorithm for different document representation and feature selection strategies

#	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 IV. Confusion matrix for the best multiclass support vector machine (MSVM) classifier

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

Finally, Table IV 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.

8. Conclusion

Legislation is usually referred in terms of documentary units, hence the difficulties in norm accessing by both citizens and legal experts, as well as the uncertainty of the impact of new rules on the legal order in terms of coherency preservation. A more analytical unit of reference, the *provision*, can be identified in order to have a more organic view of the legal system. The detection of the provision types within a legislative text is basically an intellectual activity which however can be effectively supported by automatic facilities. In this paper a module able to classify fragments of legislative texts into provision types has been presented. In particular, the approaches used to represent document features amenable for computation have been illustrated and two machine learning methodologies (Naïve Bayes and Multiclass Support Vector Machine) have been used and tested. The experiments give evidence of satisfactory results. To improve the classification accuracy, heuristics based on legislative technique rules are expected to be combined with the machine learning approaches used in this work: the aim is to introduce an exogenous knowledge (Sebastiani 2002) able to describe, within the same document, the relationships among provisions, which can make their classification easier.

Notes

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

[2] We use the bold case for d in order to stress its vectorial nature.

References

Apté, C., Damerau, F. J. and Weiss, S. M. (1994). Automated Learning of Decision Rules for Text Categorization, *ACM Transactions on Information Systems* 12(3): 233–251.

Bartolini, R., Lenci, A., Montemagni, S., Pirrelli, V., and Soria, C. (2004). Automatic Classification and Analysis of Provisions in Italian Legal Texts: A Case Study. In Proceedings of the Second International Workshop on Regulatory Ontologies.

Biagioli, C. (1991). Definitional Elements of a Language for Representation of Statutory. Rechtstheorie 11.

Biagioli, C. (1992). Law Making Environment. In Proceedings of Workshop on Legal Knowledge and Legal Reasoning Systems, Tokyo.

Biagioli, C. (1997). Towards a Legal Rules Functional Micro-ontology. In Proceedings of Workshop LEGONT '97.

Biagioli, C., and Francesconi, E. (2005). A Semantics-based Visual Framework for Planning a New Bill. In Proceedings of the Jurix Conference: Legal Knowledge and Information Systems, 103–104.

Biagioli, C., Francesconi, E., Passerini, A., Montemagni, S., and Soria, C. (2005a). Automatic Semantics Extraction in Law Documents. In Proceedings of International Conference on Artificial Intelligence and Law, 133–139.

Biagioli, C., Francesconi, E., Spinosa, P., and Taddei, M. (2005b). A Legal Drafting Environment Based on Formal and Semantic XML Standards. In Proceedings of International Conference on Artificial Intelligence and Law, 244–245.

Biagioli, C., and Turchi, F. (2005). Model and Ontology based Conceptual Searching in Legislative XML Collections. In: Proceedings of the Workshop on Legal Ontologies and Artificial Intelligence Techniques, 83–89.

Bishop, C. (1995). Neural Networks for Pattern Recognition. Oxford University Press: Oxford.

Boer, A., Hoekstra, R., and Winkels, R. (2002). MetaLex: Legislation in XML. In Proceedings of JURIX 2002: Legal Knowledge and Information System, 1–10.

Buckley, G. S. and Salton, G. (1988). Term-weighting Approaches in Automatic Text Retrieval, *Information Processing and Management* 24(5): 513–523.

Burges, C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. In Data Mining and Knowledge Discovery, Vol. 2, Kluwer Academic Publishers: Boston.

Cortes, C. and Vapnik, V. (1995). Support Vector Networks, *Machine Learning* 20, 1–25.

Crammer, K. and Singer, Y. (2002). On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines, *Journal on Machine Learning Research* 2, 265–292.

Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive Learning Algorithms and Representations for Text Categorization. In CIKM '98: Proceedings of the Seventh International Conference on Information and Knowledge Management, 148–155, ACM Press: New York, NY, USA.

Hsu, C.-W., and Lin, C.-J. (2002). A Comparison of Methods for Multi-class Support Vector Machines. IEEE Transactions on Neural Networks 13(2): 415–425.

Jensen, F. (1996). Introduction to Bayesian Networks. Springer-Verlag.
Joachims, T. (1997). A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. In Proceedings of the Fourteenth International Conference on Machine Learning, 143–151, Morgan Kaufmann Publishers Inc.
Lee, Y., Lin, Y., and Wahba, G. (2001). Multicategory Support Vector Machines. Technical Report 1043, Dept. of Statistics, University of Wisconsin.
Lewis, D. (1992). Automating the Construction of Internet Portals with Machine Learning. In Proceedings of ACM International Conference on Research and Development in Information Retrieval, 37–50.
Megale, F. and Vitali, F. (2001). I DTD dei documenti di Norme in Rete, *Informatica e Diritto* 1, 167–231.
Ng, A. and Jordan, M. (2002). On Discriminative vs Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In Dietterich, T., Becker, S. and Ghahramani, Z. (eds.), Advances in Neural Information Processing Systems 14. MIT PressCambridge, MA.
Palmirani, M. (2005). Time Model in Normative Information System. In Post-proceedings of the ICAIL Workshop on the Role of Legal Knowledge in e-Government.
Passerini, A. (2004). Kernel Methods, Multiclass Classification and Applications to Computational Molecular Biology. Ph.D. thesis, Università di Firenze: Italy.
Quinlan, J. (1986). Inductive Learning of Decision Trees, *Machine Learning* 1, 81–106.
Raz, J. (1977). Il Concetto di Sistema Giuridico. Il Mulino: Bologna.
Schölkopf, B. and Smola, A. (2002). Learning with Kernels. The MIT Press: Cambridge, MA.
Sebastiani, F. (2002). Machine Learning in Automated Text Categorization, *ACM Computing Surveys* 34(1): 1–47.
Shawe-Taylor, J., and Cristianini, N. (2004). Kernel Methods for Pattern Analysis. Cambridge University Press.
Spinosa, P. (2001). Identification of Legal Documents through URNs (Uniform Resource Names). In Proceedings of the EuroWeb 2001, The Web in Public Administration.
Vapnik, V. (1998). Statistical Learning Theory. Wiley: New York.
Weston, J., and Watkins, C. (1998). Multi-class Support Vector Machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science.
Yang, Y., and Pedersen, J. (1997). A Comparative Study on Feature Selection in Text Categorization. In Proceedings of the Fourteenth International Conference on Machine Learning, 412–420, Morgan Kaufmann Publishers Inc.