BACHELOR THESIS 2010
Credits: 9 EC

submitted in partial fulfillment of
the requirements for the degree of

BSc ARTIFICIAL INTELLIGENCE

at the UNIVERSITEIT VAN AMSTERDAM

Faculty of Science
Science Park 904
1098 XH Amsterdam

Machine Learning vs. Knowlegde Engineering in Classification of Sentences in Dutch Law

Kai Krabben [5733036]
kai.krabben@student.uva.nl

Version: June 25, 2010

UNDER SUPERVISION OF

Dr. Radboud Winkels
R.G.F.Winkels@uva.nl

Ir. drs. Emile de Maat
E.deMaat@uva.nl

LEIBNIZ CENTER FOR LAW

Faculty of Law
Universiteit van Amsterdam
Kloveniersburgwal 48
1012 CX Amsterdam

Abstract

The ultimate goal of the application of artificial intelligence in the field of legislation is to support automated modeling of sources of law. These models can help to provide fast, comprehensible and easy access to up-to-date legal knowledge to both civilians and professionals in the legal domain. A first step towards the translation of provisions into formal models is the classification of legal texts into different categories. Research in this field showed that both a machine learning (ML) approach as a knowledge engineering (KE) approach can be successfully applied to this task. This work presents a comparative study between pattern based KE techniques and a ML approach that uses support vector machines to classify sentences from Dutch law. The results suggest that at a sentence level ML techniques can achieve good results (over 90% accuracy), but that they can at best equal the results obtained by the more robust KE approach.

Contents

1 Introduction **1**

1.1 Background 1

1.2 Related Work 2

1.3 Research Question 2

1.4 Outline of the Paper 2

2 Methods **3**

2.1 Provisions 3

2.2 Data 4

2.3 Document Representation 5

2.4 Classification Algorithm 6

3 Experiments and Results **7**

3.1 Experimental Environment 7

3.2 Evaluation methods 8

3.3 Preprocessing Experiments 8

3.4 Generalisation Across Laws 8

3.4.1 Overfitting? 13

3.5 Analysis of Misclassified Sentences 13

4 Conclusion and Discussion **16**

4.1 Conclusions 16

4.2 Discussion 16

4.3 Future Work 17

1 Introduction

1.1 Background

In a modern constitutional state, all legal action is grounded within and justified by written sources of law. Any application that aims to provide access to up-to-date legal knowledge to either civilians or professionals in the legal domain *must* therefore be directly built from these sources. For machines to reason about legal knowledge, these sources have to be translated from natural language to a formal language. This process is usually performed as a combined effort of knowledge engineers and domain experts, which is both time and effort consuming. It is the ultimate goal of the application of artificial intelligence in the field of legislation to support an automated approach to the modeling of sources of law. A first step in this task is to organize the legal information by automated classification of parts of the texts into predefined concepts. In a next step, the knowledge of the classification can be used to create a formal model of the text (as different text types call for different models).

Text categorization (TC) was defined by Sebastiani [22] as *the task of assigning an electronic document to one or more pre-defined categories, based on its contents* (i.e. not on meta data). The first successful approach to solve the task of TC was to manually build classifiers based on knowledge engineering (KE) techniques. These techniques require expert knowledge to define syntactic rules (patterns) for each inidividual class that prescribe how to classify documents into different categories. In the '90s machine learning (ML) techniques became increasingly popular in classification tasks in general and also in TC. ML techniques automatically build a classifier by *learning* the characteristics of each category from a set of labeled examples[1]. In general ML bears a lot of advantages over KE. ML techniques are more flexible, less domain dependent and require less expert knowledge. A downside of ML is that it works like a black box; the reasons for classifying a certain text in a certain category can be hard to understand, whereas in KE these are given and can be explained by the domain experts. Furthermore, ML requires a sufficiently large dataset to be available, whereas KE techniques do not require any training data at all. Nevertheless, ML is nowadays considered to be the way to go in TC. In general it is more accurate and easier to implement and the downsides are usually not considered to be a problem; it is often more important that a classifier functions accurately than how this is achieved and the acquisition of enough data is most of the times an easy matter in the digitalized society that we live in.

However, the TC problem in the legal domain bears some characteristics that appeal very well to the KE approach. First of all, the amount of variation normally observed in the use of natural language is quite restricted in legal texts. Legal drafters use formal language that usually adheres to writing conventions of earlier texts and official writing guidelines; similar information is usually expressed in syntactically similar statements. Since usually only a few clear and easily observable patterns are used for each category, it is plausible that in the field of legislation KE techniques might outperform ML in the TC task. If a perfect KE classifier would be available that is equipped with all the classification patterns occurring within legal sources, it is expected that a ML approach can at best equal the results of this classifier by extracting all of the same patterns automatically. Furthermore, although KE requires more effort and the use of expert knowledge on forehand, it offers a better starting point for the modeling step of classified documents to extract the semantics of the text. Consider for example the sentence

> *By Our Minister is understood: Our Minister of Economic Affairs*

Building a formal model of this sentence is a lot more feasible if we know that the classification was based on the pattern *by x is understood y*, from which it is immediately clear what is defined and what is the definition, than from a Machine Learner that can only output the class itself. It seems almost inevitable to use some form of pattern-based

[1] More precisely, this describes a subset of Machine Learning techniques known as *supervised learning*. See [18] for an introduction into the field of machine learning and a description of the different algorithms developed within this field.

parsing to construct formal models of the texts. Work done in KE in this earlier stage will therefore save time and effort later on. Also, KE techniques can offer explanations for the choices that are made whereas ML techniques function as a black box. In systems that are used to support experts in the legal field to make important choices, these explanations could prove to be desirable or even necessary.

1.2 Related Work

Categorization of legal text is a special case of TC. It is therefore natural to assume that ML will be the best approach to solve this task, as it is in the general TC problem. Biagoli et al. [1] (and further studied in [10]) have shown that the ML approach is indeed effective in the domain of legislation. With the use of support vector machines (SVM) they classified 582 paragraphs from Italian laws into ten different categories. Several document representation and preprocessing methods were tested, leading to accuracy levels up to 92%.

Gonçalves [11, 12] and Opsomer [19, 20] applied similar methods to the domain of respectively Portuguese juridical documents from Supreme Courts and the Attorney General's Office and Belgian environmental and energy legislation. Gonçalves reached precision values above 0.9. Opsomer obtained results no higher than 0.65, which was partially blamed on the granularity of classification: classification was done on complete chapters of law, which Opsomer claimed to contain information that is too diffuse to be classified into one class.

For the reasons given in the previous section De Maat and Winkels [7] investigated the use of KE for clasification of sentences in Dutch laws. After an analysis of twenty Dutch laws fourteen categories were defined and 88 different classification patterns (phrases) to recognize them. These patterns were used to classify the sentences in eighteen other Dutch laws. Out of 592 sentences, 91% was classified correctly. Apart from the use of a KE approach, this study also differs from other related work in that classification is performed at sentence level.

1.3 Research Question

The previous section shows that both ML and KE methods yield quite good results for TC in the legal domain. The interesting question is of course if one approach can be shown to outperform the other (under certain circumstances). However, the different studies conducted in this area are hardly comparable. Different corpora have been used with legislative texts from different countries and different areas of law, along with different ways of partitioning laws in single text elements for classification and different (although overlapping) sets of possible classes. This paper describes a comparison of the two most successful studies towards TC in the legal domain, by applying the techniques used by Biagoli et al. [1] on the data set used by De Maat and Winkels [7]. The central question is:

> Does the Machine Learning approach or the Knowledge Engineering approach perform better in the classification of sentences in legal texts?

1.4 Outline of the Paper

The remainder of this paper is organized as follows. The next section describes the methods that were used to acquire, represent and process the data and the classification algorithm that was used. In section 3 an overview of the different experimental setups is presented along with the results of these experiments. The final section offers the conclusions

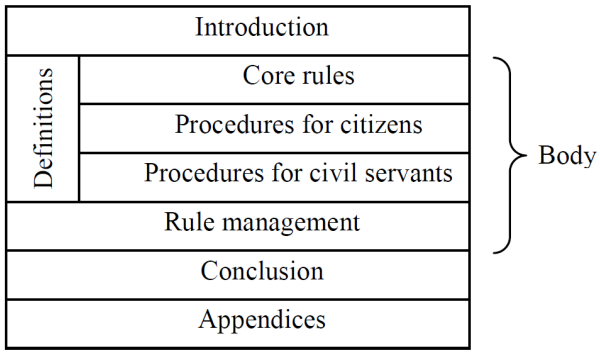

Figure 1: Overview of the types of provisions in Dutch Law.

of this project, a discussion on the use of KE vs. ML techniques in classification of legal texts and some suggestions for future work.

2 Methods

2.1 Provisions

An overview of the types of sentences or provisions in (Dutch) legislation as distinguished by De Maat and Winkels in [7] is given in figure 1. The actual legislative content is located within the *body* of the law. The core rules contain rights and duties for people and organizations within the jurisdiction of the law. Procedural rules state how the core rules should be carried out by citizens and civil servants. Definitions define a vocabulary used in the core and procedural rules. Provisions in the rule management section deal with the maintenance of the other provisions within the legal system, for example by stating an enactment date or by making textual changes in other legal sources.

Since the legislative meaning of a law is within the body of the text, these are the sentences that should be classified and eventually translated in a formal model. As explained in [7], most sentence types can be recognized by the use of specific keywords or patterns. However the majority of obligations in Dutch law is conveyed in a *statement of fact*; the text does not state what must happen, but instead simply states that it happens. For example:

> No dead bodies are interred on a closed cemetery.

Although not explicitly mentioned, the legal meaning of this sentence is not that nobody buries corpses on a closed (meaning full) cemetery (as in *no penguins fly*), but that it is by law *forbidden* to do this.

In the pattern-based classifier built by De Maat and Winkels these sentences were classified by default; if no other classification pattern was found, the sentence was classified as an obligation. De Maat and Winkels defined the following categories from analysis of twenty Dutch laws.

- **Definitions** describe the terms used in a legal source and usually follow patterns like *by x is defined y*.

- **Norms** form the actual content of legislation. They are divided into permissions (rights) and obligations (duties/prohibitions). Permissions can normally be recognized by the use of the verb *kunnen* (may). The majority of the obligations does not follow a pattern, but is conveyed as a *statement of fact* (see above).
- **Delegations** confer the power to create additional rules to some legal entity. They can be conveyed either as a permission or as an obligation (depending on whether rules can or must be created by the legal entity).
- **Penalizations** carry the punishment for violating a norm in the form of a fine or imprisonment.
- **Value assignments** assign or change the value of some concept. They are not very common except in certain legal domains such as Tax Law.
- **Application provisions** specify situations wherein some regulations do or do not apply and can be recognized by the pattern *does (not) apply*.
- **Publication provisions** specify where certain (changes in) legislation must be published.
- An **enactment date** is included in a law to describe when it's content becomes effective.
- A **citation title** title is included in most laws and can be used to refer to the law.
- **Changes** deal with textual modifications in other legal sources. Changes are further divided into insertions, deletions/repeals, replacements and renumberings. Together with the enactment date and citation title, the changes form the rule management section of a law (see figure 1).
- **Mixed types** are a combination of two patterns within one sentence, such as *The Advisory Board oversees the general affairs of the council and may advise the Minister about it* (the first part of the sentence being an obligation and the last part a permission). Because any combination of two (or more) types forms a unique mixed type, no general features can be learned or caught in patterns. Mixed types are therefore not included in any training set and therefore always misclassified.

2.2 Data

The test set used in the research of De Maat and Winkels [7] was used as the main data set for this study. This set consisted of eighteen Dutch laws, containing 588 sentences. Four of these were complete new laws, the others were amending laws dealing with changes in existing laws. Most of the laws are fairly recent (within the last few years), except for a Royal Decree from 1945. All laws except for the Royal Decree were bills, pending at parliament. The length of the files varied from only 3 up to 141 sentences. The full list of all laws is given in table 3. The majority of the sentences in the data set consisted of obligations (171) and the different kind of changes (in total 240). The set contained only one value assignment and no penalizations.

The laws were structured with MetaLex[2], an XML markup language for legal sources proposed by Boer et al. [2] and currently well on it's way to becoming the formal and de facto standard for XML in legislation. A parser was built in Python[3] to extract all sentences (marked in Metalex as *sentence* element) from the data as a string of plain text.

MetaLex also specifies specific sentences structures with an embedded list, for example:

The rapport is dated and specifies:

[2]http://www.metalex.nl
[3]http://www.python.org

a. the name of the offender;

b. the violation as well as the statutory provision that is violated;

c. the facts and circumstances under which it is determined that a violation has been committed;

d. where and when the facts and circumstances referred to under c occurred.

De Maat and Winkels in [7] and also in earlier work [6] experimented with different ways to classify lists. The best results for list classification achieved by De Maat and Winkels based the classification of the full list on the content of the list header. The assumption that it is always possible to classify a list based on it's header did not hold [7]. Another approach (also [7]) involved trying to classify a list by the header first, but classifying each of the list items separately when no classification pattern could be found. However, this approach performed less optimal than classification by the header only. Because of these difficulties the list elements were excluded from the train set. However lists were included in later test sets (see section 3.4), where the classification by header only strategy was evaluated for a ML approach.

Several sentences deal with changes in other laws. Often, old and new text is included into the modifying sentence, as in:

In article 15d, part 1, under a, "facility" is replaced by "institution".

In MetaLex, inserted or replaced texts are marked using so-called *quote* elements. Sometimes these elements contain only a small phrase (as in the above example), however they can also contain sentences or even full articles or paragraphs. When a quote element contained complete sentences, these were extracted separately as well as the containing change sentence.

All sentences were manually classified into the categories described in the previous chapter (each sentence was classified into exactly one catagory).

2.3 Document Representation

A document (sentence) needs to be represented in a way that can be managed by an automated classifier. This problem concerns the selection and representation of terms that are used to describe a single document. The most common approach in information retrieval is to select individual words as terms (using spaces and punctuation as splitting points). This is known as the Bag-of-Words model: a document is represented by the words that it contains. Several preprocessing methods can be performed on the original words to increase their use in classification:

- A *stoplist* can be used to prevent the extraction of common words like *the* or *him*;
- *Grouping of numbers* can be used to represent all numeric values by a special character;
- *Stemming* can be used to transform words to their morphological root. The Dutch version of the Snowball stemmer[4] was used in this study;
- *Conversion to lower case characters* can be considered as a special form of stemming that reduces a word to a basic form of only lower case characters.

[4] `http://snowball.tartarus.org/algorithms/dutch/stemmer`

From the selected words a vocabulary **V** is formed containing all words that occur within the dataset. Each document is represented as a high-dimensional vector of weights $w_1, ...w_{|\mathbf{V}|}$, where the weight w_i represents what information is gained from the ith term in the vocabulary in this document. Several commonly used term weighting methods were tested:

- A *binary* weight that indicates the presence/absence of the term in the document;
- A *term frequency* weight (TF) that gives the frequency of the term within the document;
- An *inverse document frequency* weight (TFIDF) that takes into account that terms occurring in many different documents are less discriminative. The weight is calculated as

$$tfidf(w,d) = tf(w,d) * log(\text{num of docs}/\text{num of docs containing } w)$$

New data will also be represented by the terms present in the vocabulary **V**; words not in the vocabulary will be ignored. In order to prevent overfitting to the train set, a *minimal term frequency* threshold can be applied over the number of times a term must occur within a class before it is put into the vocabulary.

2.4 Classification Algorithm

Support Vector Machines (SVMs) have been proven time after time to be the superior algorithm for the TC task. SVMs have been shown to perform at least as good and usually much better in text categorization than other popular algorithms such as decision trees or naive Bayes classifiers [15,22,24]. SVMs are very robust, generalize well across high-dimensional data and are easy to implement since most of the parameter tuning is done automatically and internally. Several studies showed that SVMs can successfully be applied to categorize texts in the legal domain [1,10,12,20]. Just as with general TC, SVMs outperform other classifiers within legal TC, as shown independently by Gonçalves [11,12] and Opsomer [19, 20] from classification of respectively Portuguese and Belgian legislative texts. Khoo et al. [16] showed that the superiority of SVMs over other classifiers also extends to general (non-legal) TC at a sentence level.

SVMs were developed as a binary classifier by Cortes and Vapnik [4]. The general idea behind the algorithm is to map input vectors to a very high-dimensional feature space, where the data can be optimally separated by a single hyperplane. *Optimal* in this case means that the separating hyperplane is selected with the widest possible margin to any of the training examples (known as the *Structural Risk Minimization* principle [4, 23]). Figure 2 illustrates this principle in a two-dimensional feature space. The idea can be generalized to a high-dimensional feature space and to not linearly separable training data.

A SVM is essentially a binary classifier. To extend the algorithm to a multi-label (more than two classes) classification problem, several methods have been proposed. A comparative study conducted by Hsu and Lin [14] showed that the One-against-one approach from Knerr at al [17] is the most suitable for practical use and at least as accurate as other approaches. This approach involves building a pairwise SVM for each combination of two classes (so $c(c-1)/2$ pairwise classifiers for a problem with c classes), that is trained to distinguish instances of the two classes from each other. In classification of an unknown instance, each binary SVM casts a vote for one class. The instance is classified into the class which received the most votes.

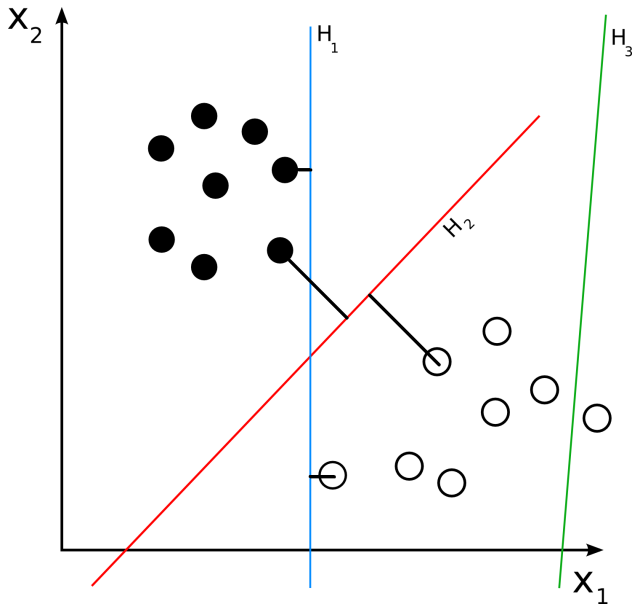

Figure 2: Illustration of the Structural Risk Minimization principle in a linearly separable 2D feature space. H1 and H2 are separating hyperplanes for this data (H3 is not). H2 is the optimal separator, since it maximizes the margin (represented with thick lines) to any of the training examples. The training instances corresponding with those margins are the support vectors.

3 Experiments and Results

This sections describes the experiments that were conducted in this study and reports the corresponding results. The first two subsections describe the experimental environment and evaluation methods that were used. In 3.3 the results of experimenting with different preprocessing methods to the data are presented. 3.4 explores how well the classifier built from train data generalizes across laws. In 3.4.1 the possibility of overfitting due to the extraction of too many terms is given (which is merely a suspicion that rose from the results) and the section concludes with an overview of hypotheses for misclassification that followed from manual examination of misclassified sentences.

3.1 Experimental Environment

The WEKA Machine Learning toolkit [13] from Waikato University[5] was implemented within a Java framework and used in the following experiments. For the SVM algorithm, the LibSVM package for WEKA [3, 9] was used with a linear kernel and default further settings. For multi-label classification this package uses the one-against-one procedure described in section 2.4.

[5] `http://www.cs.waikato.ac.nz/ml/weka`

3.2 Evaluation methods

Several evaluation methods were employed for the following experiments. The usual form of ML evaluation involves building a classification model from a train set and testing the performance of this model on another set of data called the test or validation set. Cross-validation is a method of using the same data set for both training and testing. In a k-fold cross-validation, the original dataset is partitioned into k subsets (if the distribution of data over classes is maintained within these subsets we are dealing with a *stratified* cross-validation).

Leave-One-Out (LOO) testing is a special form of cross-validation with the number of folds equal to the number of data instances; in this procedure each instance is selected for testing once and evaluated on a classifier built from the train set of all other instances. This approach makes optimal use of the amount of data that is available.

3.3 Preprocessing Experiments

Several experiments were run to determine the optimal document preprocessing settings. The original dataset was used within these experiments and evaluated through the Leave-One-Out (LOO) procedure. Mixed types were excluded from this experiment, as was the single Value Assignment since at least two instances of a class need to be in the training corpus to be able to classify it correctly (one for training and one for testing). The results are shown in table 1. Without any further preprocessing, a quite high accuracy (over 93%) is already achieved. Furthermore, the effect of any form of preprocessing tested is very small. This is in agreement with the claim made by Joachims [15] that SVMs handle most of the parameter tuning and optimal feature selection internally. Nevertheless, a small positive effect is observed from the use of a stoplist and a minimal term frequency of two (preventing the selection of words that occur only once within a class). The use of TF or TFIDF term weights, grouping, stemming and conversion to lower case characters bear no positive effect to the LOO accuracy. When combined, the use of a minimal term frequency and a stopword list increases the LOO accuracy up to 94.7%. Detailed data for classification under these otpimal settings is given in table 2. The vast majority of all sentences dealing with rule management is classified correctly. More than half of all 31 misclassified sentences were false obligations, which is by far the largest class within the data and therefore the default classification. The classifier has a lot of trouble handling definitions; almost half of the definitions is misclassified as obligation. Eight misclassifications are caused by a confusion within norms, i.e. permissions classified as obligations or the other way around.

3.4 Generalisation Across Laws

From the previous experiment we can conclude that a SVM classifier can learn to classify sentences from laws in a given dataset. The more interesting question is how well the classifier generalizes across different laws, and if it can be used to classify new (unseen) data as well. Two experiments were run to evaluate how well the learned classification rules generalize across laws.

In the first experiment, each law in the original data set was used as test set once and classified by a classifier built from all other laws in the data set. The results are shown in table 3. The first two columns contain the law and the amount of sentences it contains. The third column holds the number of sentences wrongfully classified by a classifier built from a train set that contains all sentences not in the extracted law. For comparison, the fourth column shows the number of sentences from the extracted dataset that were misclassified within the LOO evaluation described above (with optimal settings). A small drop in accuracy from the LOO to the Train/Test testing can be expected from the decreased amount of training data. This is indeed observed for the most cases. If the difference is bigger than a few instances however, it

	term weight	Stoplist	Group NRs	Stemming	Min Term Freq	Lower Case	LOO accuracy
baseline	binary	no	no	no	1	no	93.32
baseline + TF	TF	**yes**	no	no	1	no	92.29
baseline + TFIDF	TFIDF	**yes**	no	no	1	no	93.32
baseline + stopwords	binary	**yes**	no	no	1	no	94.01
baseline + grouping	binary	no	**yes**	no	1	no	92.81
baseline + stemming	binary	no	no	**yes**	1	no	92.47
baseline + MinTermFreq 2	binary	no	no	no	**2**	no	93.15
baseline + MinTermFreq 3	binary	no	no	no	**3**	no	92,47
baseline + lowercase	binary	no	no	no	1	**yes**	93.15
optimal	binary	yes	no	no	2	no	**94.69**

Table 1: LOO accuracy of different preprocessing settings on original data.

must be concluded that the file contains some instances that are unique to that file and that can not be classified from the other data. This conclusion is supported by the results of LOO testing of each data set separately, shown in the last column. This shows that for some laws (most notably Bill 30 411 nr. 2) a classifier built from the other data functions worse than a classifier based on the test set itself. Nevertheless, the results of the LOO evaluation on the full dataset are always better, indicating that the optimal classification used both common features in all data and specific features within the separate files.

In the second experiment, a classifier was built from the full original train set. Two completely new laws were used to test the classifier. Both laws were bills pending at parliament and both were very recent. The first bill (may 2010) was an amendment of other laws. The second bill (june 2010) presented a completely new law, which also brought about a lot of changes in existing laws. Both laws contain one mixed type which can not be classified correctly. Apart from the sentences, the lists in these laws were also tested and classified based on the content of the list header (as described in section 2.1). This new test set was also classified by the pattern based parser built by De Maat and Winkels [7] (the KE classifier). The results are compared in table 4. For the amending law, both the classifiers performed very well (the ML classifier even slightly better). On the second law however, the ML classifier showed worse (although still acceptable) results whereas the performance of the KE classifier remained high. The confusion matrix for the classification with the ML classifier (see table 5) shows that again all the sentences within rule management were classified correctly (as was to be expected from the classification of the amending law). Once again, the classifier had trouble dealing with definitions. Three delegations were classified as a permission, which is technically speaking not wrong but just not specific enough (since these delegation were conveyed as permissions, for example *Onze minister kan regels stellen. . . / Our Minister may create rules. . .*). The biggest proportion of mistakes comes from confusion within norms; permissions and obligations were confused with each other in over half of all the misclassifications. The results for the classification of lists was almost flawless for the KE classifier. The ML classifier also achieved good results.

classified as →	*definition*	*permission*	*obligation*	*delegation*	*publication provision*	*application provision*	*enactment date*	*citation title*	*change scope*	*change insertion*	*change repeal*	*change replacement*	*change renumbering*
definition	**8**		6										
permission		**54**	4			1							
obligation	3	4	**172**			1	1						
delegation		1	1	**17**									
publication provision			1		**5**								
application provision	1		3			**37**							
enactment date		1					**17**						
citation title		1						**3**					
change - scope									**55**				
change - insertion										**44**			
change - replacement											**111**		
change - repeal												**23**	
change - renumbering			2										**7**

Class	*In Corpus*		*Missed*	*False*	*Recall (%)*	*Precision (%)*
Definition	2%	14	6	4	57.14	75.00
Permission	10%	59	5	7	91.53	88.52
Obligation	31%	181	9	17	95.01	91.01
Delegation	3%	19	2	0	89.47	100
Publication Provision	1%	6	1	0	83.33	100
Application Provision	7%	41	4	2	90.24	94.87
Enactment Date	3%	18	1	1	94.44	94.44
Citation Title	1%	4	1	0	75.00	100
Change - Scope	9%	55	0	0	100	100
Change - Insertion	7%	44	0	0	100	100
Change - Replacement	19%	111	0	0	100	100
Change - Repeal	4%	23	0	0	100	100
Change - Renumbering	2%	9	2	0	77.78	100
Total		584	31	31	94.67	94.47

Table 2: Confusion Matrix of LOO classification on full data set, along with detailed data per class.

Extracted Law	Size	Nr Misclassified Sentences		
		Train/Test	Original LOO	Test set LOO
Royal Decree Stb. 1945, F 214	28	8	6	11
Bill 20 585 nr. 2	31	7	4	13
Bill 22 139 nr. 2	22	2	1	10
Bill 27 570 nr. 4	21	5	2	7
Bill 27 611 nr. 2	11	0	0	5
Bill 30 411 nr. 2	141	28	7	22
Bill 30 435 nr. 2	40	3	3	10
Bill 30 583 nr. A	26	1	0	5
Bill 31 531 nr. 2	3	1	1	3
Bill 31 537 nr. 2	23	0	0	4
Bill 31 540 nr. 2	7	1	1	5
Bill 31 541 nr. 2	8	0	0	3
Bill 31 713 nr. 2	7	0	0	7
Bill 31 722 nr. 2	32	9	1	7
Bill 31 726 nr. 2	78	6	3	7
Bill 31 832 nr. 2	7	1	1	2
Bill 31 833 nr. 2	4	1	1	4
Bill 31 835 nr. 2	99	7	0	7

Table 3: Results of testing each law in the dataset separately.

Test set	Size	ML approach		KE approach	
		Nr misclassified	Accuracy	Nr misclassified	Accuracy (%)
Bill 32 393 nr. 2 - sentences	71	3	95.77	4	94.37
Bill 32 398 nr. 2 - sentences	205	23	88.78	9	95.61
Bill 32 393 nr. 2 - lists	18	3	83.33	1	94.44
Bill 32 398 nr. 2 - lists	9	0	100	0	100

Table 4: LOO accuracy of different preprocessing settings on original data

classified as →	*definition*	*permission*	*obligation*	*delegation*	*application provision*	*enactment date*	*citation title*	*change scope*	*change insertion*	*change repeal*	*change replacement*	*change renumbering*	**Nr. Relevant**	**Recall (%)**
definition	**1**		2										3	33.33
permission		**19**	11	1									31	61.3
obligation		3	**47**										50	94.0
delegation		3		**2**									5	40.0
application provision			2		**7**								9	77.77
enactment date						**1**							1	100
citation title							**1**						1	100
change - scope								**16**					16	100
change - insertion									**5**				5	100
change - replacement										**71**			71	100
change - repeal											**11**		11	100
change - renumbering												**1**	1	100
mixed type									1				1	0
Nr. Retrieved	1	25	62	3	7	1	1	16	6	71	11	1	**205**	**88.78**
Precision (%)	100	76.0	75.81	66.67	100	100	100	100	83.33	100	100	100	**88.78**	

Table 5: Confusion Matrix of ML classification results for law 32398 nr. 2. Since the test set contained no value assignments or publication provisions and no instances were falsely classified into one of these classes, they are not included in this table for the sake of readibility.

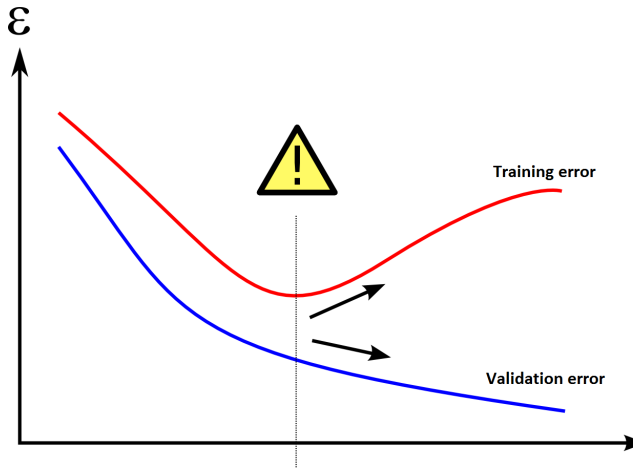

Figure 3: Overfitting in supervised learning takes place when the validation error increase while the training error decreases. Without validation on test data, it is almost impossible to notice when overfitting starts to take place.

3.4.1 Overfitting?

Joachims [15] claimed that SMVs are resistant to overfitting, the process where a classifier due to further training shows improved accuracy on the train set but decreased accuracy on new test data (see figure 3). Overfitting usually means that features specific to the domain of the train data are used to classify new data from other domains. However the drop in accuracy from the LOO evaluation on the train set to the evaluation on a new test set suggest that the trained classifier overfitted to the data. In all the above experiments, no limit was set to the number of extracted terms from the train set; between 700 and 1700 terms were extracted in the above experiments (i.e. every sentence was represented by a 700 to 1700 dimensional vector). To investigate the question whether overfitting took place, some of the experiments were rerun with a maximum number of selected terms set to 50 words per class, leading to extraction of 250 up to 500 words. The results (see table 6 show a decrease in train accuracy and increase in test accuracy from the original experiments, suggesting that overfitting indeed took place when no limit was set to the amount of selected terms. Related work done in TC trough SVMs [1, 10, 12] all used the same data set for training and testing (through cross-validation or LOO testing). None of these studies investigated the classification on completely new test sets that were not involved in the building process of the classifier. Biagioli [1] et al. reported LOO testing with usually perfect accuracy on the train set, which could indicate that the classifier is overfit to the data. Gonçalves in a preliminary study [11] reported a reduced cross-validation accuracy due to the reduction of selected features from nearly perfect (up to 99.5%) to levels between 93 and 96%. Although this is taken to be a loss in general accuracy, it can also be an indication that the high accuracy levels were a results of overfitting to characteristics specific to the data set that was used. Also Opsomer [20] in a discussion on his dissapointing results names the partial inability of a SVM classifier to generalize from train data to test data from another domain. Further research is required to support or falsify this belief.

3.5 Analysis of Misclassified Sentences

The misclassifications in the above experiments were manually studied to find hypotheses on when and why the classification went wrong. Several types of possible underlying errors for these mistakes were found and compared

experiment	Nr Correctly classified	Nr Misclassified	Accuracy (%)
LOO evaluation train set	544	40	93.15
Test set 32393 nr. 2	69	2	97.18
Test set 32398 nr. 2	187	18	91.22

Table 6: Rerun of some experiments with a maximum number of selected terms set to 50 per class.

with errors made by the KE parser. The first two types of errors were found in both the KE as the ML approach:

- **Auxiliary sentences/adverbial phrases**
  Quite a proportion of all misclassifications for both classifiers can be blamed on patterns or keywords occurring in auxiliary sentences or adverbial phrases. Consider for example the sentence

  > *Hij stelt in dat geval Onze Ministers vooraf in kennis van zijn plannen, ten einde hun mening daaromtrent te* ***kunnen*** *vernemen / He notifies Our Minister of his plans in advance, so that he* ***can*** *hear their opinion.*

  The main sentence forms an obligation, but the word *kunnen (can)* in the auxiliary sentence signals a permission. Since classifcation of a sentence always relies on the main sentence, removal of auxiliary sentences and adverbial phrases in preprocessing seems to be a promising method to improve the results for both classifiers.

- **Missed Patterns**
  As De Maat and Winkels [7] concluded in their work, whenever a distinctive pattern in the data is missed, the accuracy can fall quickly. The same goes for the ML classifier; some distinctive patterns (especially the less occurring patterns) were not found by the classifier, like the word *bevoegdheid (qualification)* that signals a permission. In a KE approach, missed patterns (when detected) can easily be added to the classifier. In ML this can be more difficult, since a manipulation of data or change in parameters is necessary to prevent the classifier from missing the pattern.

Only one type of error was found in the KE results that did not trouble the machine learner. The use of **variations of known patters** (like a change in word order, or the insertion of particles such as *also* or *then* within the pattern) are easily recognized by the flexible ML classifier, whereas in the KE approach these mistakes have to be recognized by hand and the set of classification patterns must be updated. A lot more sources for errors were found that were specific to the machine learning approach:

- **Skewness**
  Machine Learners tend to favour bigger classes and prevalent patterns over the more uncommon patterns or smaller classes, which has a general positive effect if the distribution of classes and used patterns in the train data represents the real-world distribution. A negative effect is that uncommon patterns or classes are sometimes mislcassified because of their small prior chance. For a KE classifier, smaller classes and uncommon patterns can form a problem in the manual definition of classification patterns since they are easier overlooked, but once found and added to the set of classification patterns they will not be ignored because of prior probabilities.

- **Passive sentences**
  In Dutch, the passive form is characterized by use of the verb *worden*, like it is by use of the verb *to be* in English. Since obligations tend to be written in the passive form, the use of the passive verb is linked to the obligation class. Although this helps the classifier to recognize statements of fact and therefore causes obligations to

be correctly classified a lot of the time, it also causes passive sentences in other classes to be misclassified as obligation. Since the pattern based classifies these passive sentences by default (see the description of the *statement of fact* in section 2.1, the problem is not encountered in the KE approach.

- **Subpattern**
  Some classification patterns form a subpart of another pattern. For instance the delegation pattern *may create rules* also contains the verb *may* which by itself identifies a permission. In the KE parser this is easily solved by choosing the longer classification patterns over the smaller, however a ML tends to favour the class that has the highest prior probability (most instances in the train data) which in this case is the permission.

- **Not enough data**
  The only two misclassifications within all changes were two renumberings. These sentences were actually reletterings and therefore the only two sentences that contained the word *verletterd / relettered*. Since a minimal term frequency of 2 was used, whenever one of the two reletterings was used as test set the word *relettered* was unique within the train set and therefore not extracted, nor used in classification of the sentence.

- **Keyword outside pattern**
  The ability to recognize variations of patterns that KE classifier uses, is one of the advantages of a machine learner. However in these experiments several misclassifications were caused by the recognition of a keyword used outside of the context of it's classification pattern, such as the word *toepassing/application* which in the context *is (niet) van toepassing/does (not) apply* signals an application provision, but is also often used by itself in other sentences. The inability of a pattern based classifier to recognize keywords outside of the context of the full patterns can thus sometimes be an asset, especially in a domain with little variation in the patterns that are used.

- **Focus on the wrong keywords**
  A pattern based classifier will only focus on keywords within the patterns. A ML can choose keywords by itself, which usually is a big advantage since it can discover unknown correlations. However within the previous experiments the classifier sometimes discovered correlations within the test set that do not generalize to new data (see also the previous section on overfitting; since these correlations exist within the train set they can help to improve train accuracy, but decrease test accuarcy). For example one train set might by coincidence contain a lot of permissions for some advisory board, but an advisory board by itself does not signal a permission. New data considering obligations of another advisory board might therefore by misclassified based on correlation within the train data.

- **Unknown errors**
  For some misclassifications no satisfying explanation for the cause of the error could be found, which shows the downside of black box Machine Learning techniques. Since the source of errors is unclear, they are hard to correct. For example the sentence:

  > *Er zal tevens rekening gehouden worden met de wenselijkheid dat verschillende in de maatschappij levende opvattingen met betrekking tot een aandachtsgebied in een adviesraad zijn vertegenwoordigd / The desirability that different opinions within society with regard to a certain area are represented in an advisory board will be also taken into account.*

  was classified as a permission while no permission keywords can be found within the sentence. Probably the classifier found a word within this sentence that by coincidence also occurred in some permissions in the train set, however manually studying the test set next to each unknown error is very time and effort consuming. In a pattern based classifier on the other hand, errors can easily be explained and usually also fixed by adding a new pattern to the classifier.

4 Conclusion and Discussion

4.1 Conclusions

It was shown that Machine Learning can successfully be applied to the classification of legal documents at sentence level. A classifier was built with SVMs on the same dataset that De Maat and Winkels [7] used to test their pattern based (KE) classifier. The ML classifier performed slightly better than the KE classifier on the original data. However, it must be noted that the ML classifier is built from the original dataset, whereas the KE classifier was built from patterns found in another dataset (so for the KE classifier this was already a test of generalization). Also, in the experiment of Maat and Winkels the mixed types (which are misclassified by default) were included, whereas in the ML experiment they were left out.

Both classifiers were also tested on a completely new set with equivalent results. Here, both classifiers showed nearly optimal accuracy within the rule management section, but the KE classifier greatly outperformed the ML classifier on core rules. Analysis of the misclassified sentences showed that the ML approach is prone to more errors and that the source of these errors is a lot more vague than with the pattern based classifier. The only advantage of the ML classifier that followed from this analysis was that it is better at handling small changes in writing conventions or guidelines that causes variations in the used patterns. However, larger changes that result in completely new changes are expected to be missed by both classifiers. In general a pattern based classifier is considered to be more robust in the categorization of legal documents at a sentence level.

The method to handle lists used in the KE approach of [7] was shown to be applicable for the ML approach as well. Although not very extensively tested, the results of classification of lists in the new data seemed to function well. The classifier used in this classification was build from only sentences (i.e. not lists) in the original dataset. This shows that lists, apart from a different structure, in general follow the same writing conventions and guidelines as sentences.

4.2 Discussion

In [22] Sebastiani reported an accuracy threshold of 90% in the general TC task for both KE and ML techniques (based on the state-of-the-art in the late '90s). Both approaches have been proven to be able to cross this threshold (in both this and in ohter, earlier mentioned, work), which indicates that the legal domain indeed, as claimed in the introduction of this paper, bears some characteristics that make it extra suitable for the TC task.

Indications were found that unlimited selection of terms can cause overfitting on the train data; further investigation is required to verify these suspicions. The key point here is to define the domain of legal texts that the classifier will be used for. A classifier trained on tax laws might overfit to specific phrases in this area and therefore generalize badly to other laws. If however this classifier is intended to be used only for the classification of other tax-related laws, learning these specific phrases is actually a good thing. A good classifier is built from a data set that resembles the full range of texts that is has to be able to classify. In a KE approach the generalization is left to human decision on what patterns extend to other domains and what patterns do not. Although the risk of overfitting still exists from manual selection of too specific patterns, the use of commons sense and a little bit of domain knowledge should be enough to prevent most of these mistakes.

The question remains whether a pattern-based KE approach or a ML approach through SVMs works best for the categorization of legal documents. The results of this study shows that SVMs perform nearly perfect in the classification of rule management sentences and also in the distinction between rule management and core rules, but that the

pattern-based classifier is more robust in the classification of (unseen) core rules.

Another consideration that has to be taken into account is the textual level at which documents are classified. In this study classification at sentence level was considered. At this level, the manual recognition of patterns is very feasible. It is not clear if this still goes for classification at paragraph level, or if this becomes to complex. In Dutch law, information is written down quite efficiently. Almost no sentence can be found that does not convey relevant legal information. This could be characteristic to Dutch legal texts; Opsomer [20] discussed that disappointing results in the classification of complete chapters in Flemish laws (written in Dutch) could be (partially) blamed on the fact that these law objects contained information too diffuse to classify into one chapter (his claim is supported by a negative correlation between classification accuracy and the size of law objects). When the goal of automated modeling of law is considered again, it is thus (at least for Dutch law) natural to conclude that this modeling and therefore the classification has to take place at a sentence level, at which the ML performance at best equals that of the KE approach. Other applications for legal TC could be thought of that might suffice with classification at a higher (paragraph, article) level, such as high level summarization or division of legislative texts. At this textual level, the performance of KE techniques might prove to be more difficult. However the results of Opsomer [20] indicate if the granularity becomes too coarse, ML can no longer yield satisfying results. At a paragraph level however, classification with SVMs has been applied successfully [1].

Another element to be taken into consideration in this discussion is the maintenance of the classifier. An advantage of the KE approach in daily maintenance is that the system can easily be improved when misclassifications because of unknown patterns are recognized, by adding this pattern to the classifier. In ML these instances can be added to the train data to rebuild the classifier, but this approach is less robust in that it can not guarantee that the pattern will now be recognized in other sentences.

Although legal drafters tend to adhere to writing conventions, these conventions can change over time. It is also possible (although undesirable) that the official writing guidelines are changed radically. For small changes in writing conventions or guidelines, a ML classifier has the advantage of generalizing to new variations of old patterns. As for greater changes that involve the use of completely new patterns, the ML approach requires the construction of a new train set representable to the current distribution of sentences used in sources of law, whereas the patterns for the KE have to be recreated. For changes in writing conventions this would involve manual study of a similar representative dataset. For changes in writing guidelines however a study of the new guidelines should be enough to understand what new patterns are expected to be used. Within the last category of changes, another advantage of the KE approach is that the classifier can be updated before the first law under the new guidelines is published (and therefore be directly applied to new laws) whereas the ML classifier can only be updated after new data has become available.

Obligations were the largest class within the data. Therefore the SVM classifier classified statements of fact by default as an obligation. Although this is the correct behavior, it is not clear whether the classifier based this decision on the fact that the obligation class is the largest and therefore default class or because it indeed learned that the statement of fact is recognized by the lack of other occurring keyword. It is therefore unclear what would happen if a train set was used with another class larger than the obligations.

4.3 Future Work

This work gave rise to several suggestions for future work. First and foremost, the desired textual level of classification should be considered very carefully with respect to the intended use of the classified text. For formal modeling of Dutch law, classification at sentence level seems to be the way to go. For future applications or domains where classification at a larger level (paragraphs, articles) is required, a comparative study between KE and ML for classification at these

levels would be helpful to support the directions of future research. Further investigation in differences and similarities between legal sources from different countries and in different languages could give insight in to which extent the optimal classification granularity differs between languages and countries. Comparative research to legislation in different languages would also be a great asset in the evaluation of work done in this line of research. In the current situation it can be unclear whether different results are caused by the use of differences in the employed methods or in the underlying properties of the data.

Research into which features are general and which are domain specific could help to put extra constraint on the term selection and thereby improve the generalization across different laws. The possibility of SVMs overfitting to the data due to excessive term selection is also worthy of further investigation, so that this suggestion can be either falsified or confirmed.

Sentences that fall within rule management were practically all classified correctly. Confusion within core rules usually came down to false norms (i.e. core rules misclassified as permission or obligation). Perhaps ML could be employed as an efficient high level classification into core rules and rule management as a first step in the classification of legal sentences, before a more low level classification is employed. This would follow a hierarchical approach as proposed by Opsomer [19, 20].

Furthermore, the cause of errors that followed from analysis of misclassified sentences in section 3.5 and possible solutions to prevent them are worthy of further investigation. One very promising approach seems to be the removal of auxiliary and adverbal phrases before classification. Although this will require another preparsing step to recognize them, the distinction between primal and auxiliary sentences and the identification of adverbial phrases could also support the creation of formal models from classified instances. The extra effort put in this step will therefore be an investment into later work (which is in general a solid argument to prefer KE over ML in this task, even when both classifiers perform equally).

As mentioned, the majority of laws being published deal with insertions, deletions or replacements of text in other laws. Sentences containing these changes showed almost perfect precision and recall with the ML classifier. This result could already be applied to actually execute these textual changes in digital databases of legal sources. Earlier work by De Maat et al. [8] showed that the MetaLex XML markup for legislation [2] can successfully be used to detect references to articles, paragraphs or full laws. Phrases that are inserted or replaced are also marked by so-called *quote* elements. From these elements it is essentially a simple matter of implementation to update the laws in a database according to these changes, so that civilians or professionals in the legal domain can have access to up-to-date legal knowledge. Systems for editing existing laws based on given amendments have already been constructed [5, 21]. In combination with accurate detection of rule management sentences within laws, this would form a powerful tool in the consolidation of legislative documents. Again, work done in this area will prove helpful in the translation of classified sentences into formal models. The few change sentences that are not correctly classified are usually mixed types that for instance both renumber one article and insert another. To correctly handle these sentences, they should be divided into different subtypes (common combinations of categories) and included in the set of classification categories (this goes for both a ML as a KE approach).

References

[1] C. Biagioli, E. Francesconi, A. Passerini, S. Montemagni, and C. Soria. Automatic semantics extraction in law documents. In *ICAIL '05: Proceedings of the 10th international conference on Artificial intelligence and law*, pages 133–140, New York, NY, USA, 2005. ACM.

[2] A. Boer, R. Hoekstra, R. Winkels, T. van Engers, and F. Willaert. Metalex: Legislation in xml. *Legal Knowledge and Information Systems. Jurix*, pages 1–10, 2002.

[3] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[5] E. de Maat, S. van de Ven, R. Winkels, and T. van Engers. Automated handling of amending documents and resulting consolidations. In *Proceeding of the 2009 conference on Legal Knowledge and Information Systems: JURIX 2009: The Twenty-Second Annual Conference*, pages 116–125. IOS Press, 2009.

[6] E. de Maat and R. Winkels. Automatic classification of sentences in dutch laws. In *Legal Knowledge and Information Systems: Jurix 2008: The Twenty-first Annual Conference*, page 207. Ios Pr Inc, 2008.

[7] E. de Maat and R. Winkels. A next step towards automated modelling of sources of law. In *ICAIL '09: Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 31–39, New York, NY, USA, 2009. ACM.

[8] E. de Maat, R. Winkels, and T. van Engers. Automated detection of reference structures in law. *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, 152:41, 2006.

[9] Y. EL-Manzalawy and V. Honavar. *WLSVM: Integrating LibSVM into Weka Environment*, 2005. Software available at `http://www.cs.iastate.edu/~yasser/wlsvm`.

[10] E. Francesconi and A. Passerini. Automatic classification of provisions in legislative texts. *Artif. Intell. Law*, 15:1–17, March 2007.

[11] T. Gonçalves and P. Quaresma. A preliminary approach to the multilabel classification problem of Portuguese juridical documents. *Progress in Artificial Intelligence*, pages 435–444.

[12] T. Gonçalves and P. Quaresma. Is linguistic information relevant for the classification of legal texts? In *Proceedings of the 10th international conference on Artificial intelligence and law*, ICAIL '05, pages 168–176, New York, NY, USA, 2005. ACM.

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

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

[15] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. 1998.

[16] A. Khoo, Y. Marom, and D. Albrecht. Experiments with sentence classification. In *Proc. of Australasian Language Technology Workshop*, pages 18–25. Citeseer, 2006.

[17] S. Knerr, L. Personnaz, G. Dreyfus, J. Fogelman, A. Agresti, M. Ajiz, A. Jennings, F. Alizadeh, F. Alizadeh, and J. Haeberly. Single-layer learning revisited: A stepwise procedure for building and training a neural network. *Optimization Methods and Software*, 1:23–34, 1990.

[18] R. Michalski, J. Carbonell, and T. Mitchell. *Machine learning: An artificial intelligence approach*. Morgan Kaufmann Pub, 1986.

[19] R. Opsomer. Automatisch classificeren van milieujuridische teksten, 2009.

[20] R. Opsomer, G. De Meyer, C. Cornelis, and G. Van Eetvelde. Exploiting properties of legislative texts to improve classification accuracy. In *Proceeding of the 2009 conference on Legal Knowledge and Information Systems*, pages 136–145, Amsterdam, The Netherlands, The Netherlands, 2009. IOS Press.

[21] M. Palmirani and R. Brighi. Norma-system: A legal document system for managing consolidated acts. In *Database and Expert Systems Applications*, pages 295–314. Springer, 2002.

[22] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.

[23] V. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 2000.

[24] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49. ACM, 1999.