Custom Text Weighting on Law Area Classification

TxMM 2018

R.M.W. Kluge
Radboud University
s4388267
ruben.kluge@student.ru.nl

ABSTRACT

Categorizing law documents into their corresponding categories manually is an exhaustive task. Text classification is not new; there are several studies showing that Support Vector Machines (SVMs) with TF-IDF and Naive Bayes approaches do a decent job in text classification. By looking at parts of the law documents, we try to improve classification scores. Because we know that introductory text of dutch law documents of *Rechtspraak.nl* contain valuable information that is relevant for classification, we investigate whether applying extra (heavier) weighting to this introductory text will improve classification performance. Each document can have multiple law categories attached. Because SVMs cannot assign multiple labels (multi-label classification) out of the box, a One-Versus-Rest (OVR) scheme is used. This also solves the problem with overlapping categories. This approach seems to improve performance (F1-score) significantly, but with a very small percentage (~0.22%).

KEYWORDS

Law Classification, Text Classification, Bag-of-Words, TF-IDF, Linear SVM, One-Versus-Rest, Text Weighting, Supervised Learning

1 INTRODUCTION

With increasingly growing court cases and different resources to grab information from, it is a valuable addition to have these resources automatically classified in their corresponding categories. For this project we got assigned a classification problem of law documents from Legal Intelligence. Legal Intelligence wants to classify law documents into categories by classifying on textual data. The data that is being used is public property of the dutch Court (Rechtspraak.nl), and consists of lawsuits and their verdicts. Each document has been assigned one or more law category (labels), and it is our task to assign these documents their corresponding labels. Although the documents consists of XML data with multiple information fields, we should solely use the document text as our information source. This paper will go into the details of how we can improve the text classification by amplifying parts of the document we think that are relevant.

2 RESEARCH QUESTION

After analyzing some documents, it becomes clear that nearly every document starts with crucial information that could determine the law area of that particular document. Information regarding the location of the sitting, which tribunal it addresses, but sometimes also small summaries of previous sittings seems to be listed at the introductory text of each document. It would be interesting to see whether this information contributes more to classification than the rest of the text. Bruninghaus [2] already suggested to focus on smaller sentences of the document instead of classifying complete documents, which will add knowledge during parsing of documents. Our research question is thus:

Would heavier weighting on introduction text improve the classification score of an SVM algorithm on dutch law category classification?

2.0.1 *Hypothesis.*

We test this research question with a one-sided hypothesis, which will show whether our experimental condition has a significant higher mean F1-score than our baseline condition:

$H_0 : \mu_{baseline} \geq \mu_{experiment}$
$H_a : \mu_{baseline} < \mu_{experiment}$

3 METHODS

We choose to tackle this problem with a bag-of-words unigram approach. This means that every word in the text is considered to be a feature in the classifier. Bigram approaches were also considered, but due to limitations in computing resources this could not be tested [1].

Research regarding classification of law documents have already been done with Naive Bayes and C4.5 approaches [9]. However, SVM seems to outperform Naive Bayes and Neural Networks (perceptron and three-layers neural networks including a hidden layer) on category classification [4]. Further investigation of neural networks found that these neural networks could be superior towards SVM approaches [7]. We still stick to SVM because of its simplicity and scalability.

The dataset contains multi-class labels, where each document can be assigned to multiple labels. Because standard SVM approaches are not multi-class, and are not able to assign multiple labels, we choose the One-Versus-Rest (OVR) approach. This means that for each class that can be assigned, there is a need for a separate SVM classifier. Later on we will discuss this pipeline in depth.

The distribution of categories in the dataset can be found in Figure 1 and Table 1. In this figure, law areas with less than 150 entries are left out for ease of visualization. The left out law areas are: *Internationaal privaatrecht, Bestuursstrafrecht, Intellectueel-eigendomsrecht, Goederenrecht, Aanbestedingsrecht, Europees civiel recht, Mededingingsrecht, Europees bestuursrecht, Strafprocesrecht, Internationaal publiekrecht, Internationaal strafrecht, Penitentiair strafrecht* and *Europees strafrecht*. For each of the 29 law categories, it is the task of the classifier to assign the correct labels to the documents. We see that this dataset is highly unbalanced, and that most documents contain the *Bestuursrecht* label.

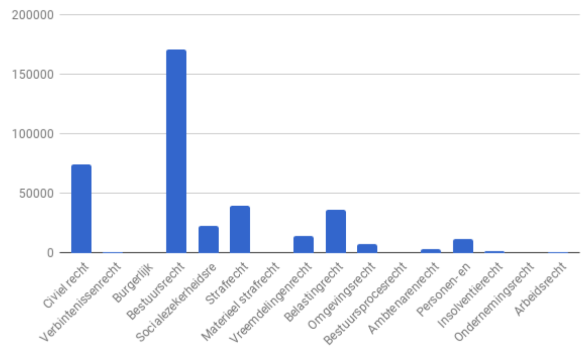
[1]A bug between Joblib & Pickle made multiprocessing on computing clusters with large datasets impossible.

Table 1: Data distribution

Law area	Frequency
Civiel recht	74444
Verbintenissenrecht	495
Burgerlijk procesrecht	282
Bestuursrecht	170266
Socialezekerheidsrecht	22962
Strafrecht	39635
Materieel strafrecht	157
Vreemdelingenrecht	14456
Belastingrecht	36697
Omgevingsrecht	7601
Bestuursprocesrecht	232
Ambtenarenrecht	3586
Personen- en familierecht	12155
Insolventierecht	1448
Ondernemingsrecht	346
Arbeidsrecht	709

For our experiment, we consider the first 10% of the text to contain introductory content. Our baseline are the original documents, whereas our experimental condition will duplicate the introductory text two times.

Figure 1: Data distribution



3.1 Pipeline

The pipeline for this experiment consists mostly of preprocessing and preparation steps in order to fulfill the promise of multi-class and multi-label classification. All steps (except preprocessing) are functions of the scikit-learn python package [8]. The current pipeline is as follows:

(1) Preprocessing
(2) CountVectorizer
(3) TF-IDF Transformer
(4) Support Vector Machine (SVM)
(5) One-VS-Rest Classifier

The experimental manipulation regarding weighting happens after the first *Preprocessing* step, as all documents get transformed and normalized in some way in later steps.

3.2 Preprocessing

The dataset that was given consists of 1.8GB of XML files (~290.000). Each file corresponds to one entry in the rechtspraak.nl database. The XML files contain a lot of extra metadata. Because we want to classify on just textual data, this other metadata is excluded. The metadata parameters we are interested in are the identifiers *Uitspraak* and *Rechtsgebied*. These identifiers correspond to our data and labels respectively.

It is noted that some documents do not even contain the label field *Rechtsgebied*. These documents are being discarded, as there are no other indicators that could determine the documents' law area for supervised learning. Other problems with the dataset are documents that do not have a subject, or faulty XML tags which causes corrupt data. These documents are discarded as well.

Other preprocessing methods such as analyzing Part of Speech (PoS) tags and lemmatization have been considered. PoS tagging allows us to have a semantic representation of words, which will help in classification. We have investigated FrogNLP [1] to process this. FrogNLP is a widely used "integration of memory-based natural language processing (NLP)" [1] for the Dutch language. The *Chunker* of FrogNLP is an intermediate step between PoS tagging and full parsing. This enables us to extract specific sentence information such as *B-NP*, *I-NP*, *O*. PoS tags like *O* could immediately be discarded, as these are symbols. The problem that occurs is that scikit's bag of words approach does not allow multiple features per word. This is because the vectorizers of scikit (DictVectorizer, FeatureHasher, CountVectorizer) do not account for bindings of multiple features (ex: text and PoS tags). A solution could be to flatten these features into one, so multiple features ('This, 'B-NP') could be flattened to one feature ('This_B-NP'). This way, we will still have the information regarding the semantics of words in our bag-of-words. For example, the dutch word 'sla' could mean 'sla_B-VP' (punching) or 'sla_B-NP' (salad). Unfortunately, FrogNLP takes around a minute per document. With 285.000 documents in the dataset to be analyzed, this is not a feasible option.

3.3 CountVectorizer

The second step in the pipeline is the *CountVectorizer*. This function converts documents into matrix representations. Each vector entry in the matrix is determined by splitting a document into words. The documents are splitted by the *CountVectorizer* by selecting strings that have two or more alphanumeric characters, which means that special characters are being ignored. We also make sure that all words are transformed into lowercase format. This matrix forms the foundation of our bag-of-words model.

3.4 TF-IDF Transformer

The third step in the pipeline is the *TfidfTransformer*. Using term frequency-inverse document frequency (TF-IDF) with a bag-of-word approach is a widely used practice in NLP [11]. TF-IDF is a way to weight text by taking into account the word distribution in the corpus. A word gets a higher TF-IDF value when a word appears often in a document (term frequency), but less often in the document corpus (inverse document frequency). Words that are occurring often in both documents and the document corpus (ex: stopping words) gets assigned less weight. The advantage of this is that less informative words get less weight, whereas less occurring informative words gets boosted weights.

The *TfidfTransformer* takes a word frequency matrix as input and will calculate its TF-IDF representation using weights. Default L2 normalization is applied to normalize word vectors, and Laplace smoothing is applied to prevent 'dividing by zero' errors.

Other approaches that include the usage of the newer BM25 instead of TF-IDF have been considered. However, [6] concludes that term frequency transformations such as BM25 show no remarkable effect on classification, and that an SVM classifier seemed 'state of the art in this domain'.

3.5 SVM & One-Versus-Rest (OVR)

The classifier where we test our custom weighting with is the *Support Vector Machine (SVM)*. Because our dataset is so large, we take an SVM that easily scales with the amount of data. As stated in [4], linear SVMs are fast in both training and evaluation, and are accurate text classifiers. Also, linear SVMs scale better to large amounts of data.

Because SVMs are binary classifiers, we utilize a One-Versus-Rest classifier scheme to perform multi-label classification (also called the binary relevance method). This means that for each class, a separate SVM is produced to solve that particular class. In our dataset this means the scaffolding of 29 SVMs through an OVR scheme. Another reason to utilize a OVR scheme is because there can be overlapping between categories. When using only one SVM, the decision boundaries can be influenced by this overlapping of categories, which results in lower classification scores.

SVM supports the same properties for classifying text as our bag-of-words approach. This approach also has a high dimensional feature space, few irrelevant features and sparse instance vectors (most documents have a vector that only a few entries that are not zero) [5]. Also, "SVMs eliminate the need for feature selection, making the application of text categorization considerably easier". The need for feature selection does not apply to SVM text classification techniques. It has been investigated that a classifier which only uses words that are the least to contribute to the classification performance, still have a better performance than when performing a random action (chance level) [5].

4 RESULTS

For each experiment that is run, 300 stratified crossvalidations are performed against each dataset, with a test size of 30%. For evaluation of scores, an F1-weighted measure is used. F1-weighted calculates the metrics for each label, and averages this by considering the amount of support per class. Classes with lower support have less influence on the averaged F1-score. The F1-score of the baseline experiment is 0.96055, where the F1-score of the experimental condition is 0.96283. The increase of the experimental condition in performance is minimal (~0.22% increase), but because we have 300 crossvalidations to support these results, this difference can be big enough to be significant.

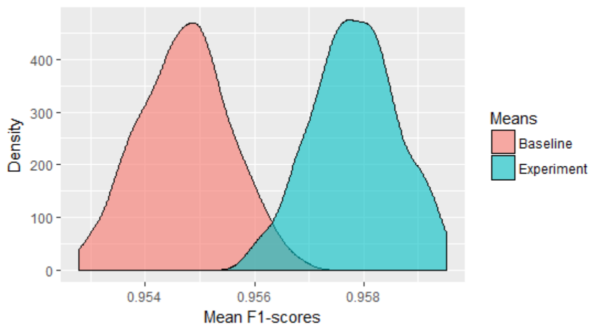
Table 2 shows us the individual class results for the experimental condition. Classes with less than 20 supporting queries are left out in the table for ease of visualization. Most of these left out classes have an F1-score of 0.00, because there is too less training data available for those classes. It is observed that classes that do have many training samples and thus high support generally have a better performances.

4.1 Significance

We are testing significance on performances. Because a performance score can never exceed 100%, this data is not normally distributed. A widely used statistical test that is being used for classification results is the Wilcoxon signed-rank test [10]. This is a non-parametric test which does not assume variance homogeneity. The samples of our results are *paired*, because the pseudorandomness factor is used to 'randomly' generate the splits. This causes every run of 300 crossvalidations to come up with the same 'random' indexes of splits after every re-run.

The Wilcoxon signed-rank test calculates a p-value of $< 2.2e-16$. This observation rejects our null hypothesis (p-value < 0.05); the mean F1-score of the baseline condition is significantly lower than the mean F1-score of the experimental condition.

Figure 2: Distribution of results



5 CONCLUSION & DISCUSSION

Weighting introduction texts heavier on dutch law documents show significantly better results in comparison to the normal weights (baseline). However, these mean differences are very small (~0.22% increase in performance, to an F1-score of 0.9628). There is much more benefit to be gained from other measures like Part of Speech tagging or semantic analysis.

Upon analysis of results, we found out that some classes are superclasses of other classes. It was found that the class label *Bestuursrecht* is a superclass of all the classes *Socialezekerheidsrecht, Vreemdelingenrecht, Belastingrecht, Omgevingsrecht, Bestuursprocesrecht, Ambtenarenrecht, Bestuursstrafrecht* and *Europees bestuursrecht*. In Table 3, we can see that the frequency of every law area corresponds with the frequencies in Table 1, which suggests that *Bestuursrecht* is a superclass.

Because we did not have any knowledge about law areas, we assumed that each label was an independent law area with no hierarchy. However, because of multi-label classification there exists a separate classifier for each class, so overlap between super- and subclasses do not cause interference in performances.

Another point that should be made is that we do not check whether each document gets assigned all of their labels. Instead,

Table 2: Individual class results

Category	Precision	Recall	F1-score	Support
Aanbestedingsrecht	0.00	0.00	0.00	28
Ambtenarenrecht	0.70	0.64	0.67	1034
Arbeidsrecht	0.75	0.20	0.31	208
Belastingrecht	0.99	0.99	0.99	11095
Bestuursprocesrecht	1.00	0.03	0.06	68
Bestuursrecht	0.99	1.00	1.00	51128
Bestuursstrafrecht	1.00	0.15	0.26	33
Burgerlijk procesrecht	0.40	0.07	0.12	89
Civiel recht	1.00	0.98	0.99	22232
Insolventierecht	0.81	0.66	0.73	441
Intellectueel-eigendomsrecht	0.50	0.08	0.13	26
Internationaal privaatrecht	0.00	0.00	0.00	24
Materieel strafrecht	0.00	0.00	0.00	38
Omgevingsrecht	0.79	0.72	0.75	2315
Ondernemingsrecht	0.90	0.69	0.78	91
Personen- en familierecht	0.87	0.91	0.89	3643
Socialezekerheidsrecht	0.82	0.74	0.78	6930
Strafrecht	0.99	0.99	0.99	11940
Verbintenissenrecht	0.25	0.01	0.01	145
Vreemdelingenrecht	0.96	0.98	0.97	4358
avg / total	0.97	0.96	0.96	115917

Table 3: Documents containing the label 'Bestuursrecht'

Law area	Frequency
Bestuursrecht	170266
Socialezekerheidsrecht	22962
Vreemdelingenrecht	14456
Belastingrecht	36697
Omgevingsrecht	7601
Bestuursprocesrecht	232
Ambtenarenrecht	3586
Bestuursstrafrecht	123
Europees bestuursrecht	21

performance is calculated whether an individual class belongs to a document correctly.

6 FUTURE WORK

In the future, we can look at different tests in order to get greater performance improvements. These tests can include testing with different (heavier) weights on introductory text, or tests to define and extract introductory text with more precision.

A next step could be to improve other algorithms like Daniels [3] with our approach. Daniels describes a method to extract semantic information from law documents themselves. By providing these algorithms with prior knowledge like law category, we can influence the extraction of semantics based on our predicted law category.

REFERENCES

[1] Antal van den Bosch, Bertjan Busser, Sander Canisius, and Walter Daelemans. 2007. An efficient memory-based morphosyntactic tagger and parser for Dutch. *LOT Occasional Series* 7 (2007), 191–206.

[2] Stefanie Brüninghaus and Kevin D Ashley. 1999. Toward adding knowledge to learning algorithms for indexing legal cases. In *Proceedings of the 7th international conference on Artificial intelligence and law*. ACM, 9–17.

[3] Jody J Daniels and Edwina L Rissland. 1997. Finding legally relevant passages in case opinions. In *Proceedings of the 6th international conference on Artificial intelligence and law*. ACM, 39–46.

[4] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. 1998. Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management*. ACM, 148–155.

[5] Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. *Machine learning: ECML-98* (1998), 137–142.

[6] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng. 2006. Some effective techniques for naive bayes text classification. *IEEE transactions on knowledge and data engineering* 18, 11 (2006), 1457–1466.

[7] Larry M Manevitz and Malik Yousef. 2001. One-class SVMs for document classification. *Journal of Machine Learning Research* 2, Dec (2001), 139–154.

[8] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, Oct (2011), 2825–2830.

[9] Paul Thompson. 2001. Automatic categorization of case law. In *Proceedings of the 8th international conference on Artificial intelligence and law*. ACM, 70–77.

[10] RF Woolson. 2008. Wilcoxon Signed-Rank Test. *Wiley encyclopedia of clinical trials* (2008).

[11] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. 2007. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*. ACM, 197–206.