PREDICTING LEGAL PROCEEDINGS STATUS: AN APPROACH BASED ON SEQUENTIAL TEXTS

A PREPRINT

Felipe Maia Polo*
Department of Statistics
University of São Paulo
São Paulo-SP, Brazil
felipemaiapolo@gmail.com

Emerson Bertolo
Tikal Tech
São Paulo-SP, Brazil
emerson@tikal.tech

Itamar Ciochetti
Tikal Tech
São Paulo-SP, Brazil
itamar@tikal.tech

January 27, 2020

ABSTRACT

Machine learning applications in the legal field are numerous and diverse. In order to make contribution to both the machine learning community and the legal community, we have made efforts to create a model compatible with the classification of text sequences, valuing the interpretability of the results. The purpose of this paper is to classify legal proceedings in three possible status classes, which are (i) archived proceedings, (ii) active proceedings and (iii) suspended proceedings. Our approach is composed by natural language processing, supervised and unsupervised deep learning models and performed remarkably well in the classification task, furthermore we had some insights regarding the patterns learned by the neural network applying tools to make the results more interpretable.

***K*eywords** Law · Process · NLP · Classification · Status · Neural Networks

1 Introduction

Machine Learning is present in many areas and is capable of performing the most diverse tasks with quality. One area that is already undergoing major changes and where there is still much room to work in the coming years is the area of law and justice. Many companies and researchers are developing new technologies to make legal processes increasingly efficient, creating great value for firms and consumers, especially democratizing services in developing countries. Law is a very wide area regarding its sub-areas and tasks, however Machine Learning applications have been proving to be very versatile, making a good deal in many of them. Examples of notable applications in the field of law would be reviewing documents, making text-based classifications or anticipating legal outcomes. A recent review paper [1] provides insights on how Machine Learning can relate to various legal tasks, and the author highlights the importance that smart tools have gained in recent years assisting legal professionals.

In this work we make extensive use of natural language processing (NLP) and machine learning tools to classify legal proceedings. Applications of NLP in the legal context are often challenging because legal texts are rhetoric, directed to persuasion and rarely descriptive, using figures of speech and other compositional techniques that challenge and twist a hypothetical "plain" and "neutral" meaning of terms, expressions and phrases. Therefore, our efforts were directed also to develop a classification model about a concrete fact – the status of legal proceedings according to a practical view –, disregarding theoretical discussions and queries on law about the nature of this status. We believe that the major contribution of this work is precisely the way we solve an important problem, described better in Section 3, combining several types of techniques to analyze sequences of texts in chronological order, which are so common in the legal context. The results obtained were satisfactory both in terms of classification performance and interpretability, which also brings importance to this work.

*Webpage: https://felipemaiapolo.github.io/

2 Related work

In recent years there has been a great advance regarding the popularization of methods for extracting relevant information from legal documents, and in the literature there are basically two classes of data to work with the prediction of legal decisions: (i) structured data and (ii) unstructured data, specifically texts. Structured data is currently scarce compared to the mass of unstructured data, especially in the legal area. Then, most of the time it is necessary to resort to the natural language processing tools for the extraction and use of information from legal documents. It is also possible to make a paradigm distinction between the use of classic Machine Learning tools and Deep Learning tools. The basic distinction between the two classes of models is that when using classic tools (Decision Trees, SVM, Naive Bayes, among others) it is necessary to do feature extraction in a less data-driven setting opposing to Deep Learning framework, which often can make the latter approach more powerful, specially when working with unstructured data. Although there are some efforts to apply machine learning in the legal world, there have not been any to solve a problem similar to ours – as far as we know –, then we are going to talk about some applications that inspired us.

The authors of [2] make use of natural language processing tools to extract features such as N-Grams and Topics and then perform a binary classification task using Support Vector Machines (SVM) on whether cases referred to the European Court of Human Rights (ECHR) contain, in its report, any violated human rights article - the most optimistic accuracy rate was 84%. Another example of work using classical tools to predict the outcome of court decisions is [3]. The authors were intended to perform three main prediction tasks that relate to cases judged by the French Supreme Court: (i) predict the legal area of a case, (ii) predict the court's decision based on the case description and (iii) estimate when the case description and a decision were issued. Results were 0.9 F1 score in the prediction of the legal area of a case, 0.96 F1 score in the prediction of a case decision and 0.76 F1 score for the third task. The methodology was composed by Bag of Words and Support Vector Machine (SVM) models together.

When choosing to use the Deep Learning tools, it is not necessary to extract features beforehand, the classification model itself does so endogenously and optimally. Some examples of such tools are Recurrent Neural Networks and Convolutional Neural Networks. If the amount of data is satisfactory and the computational power is not a limitation, a good use of these tools tend to give a better result than those achieved by traditional tools. A recent Brazilian study [4] makes use of Convolutional Neural Networks to classify documents analyzed by the Brazilian Supreme Court (STF), achieving significant results. The classification proposed by the authors is made for six different classes, which were not translated to english by the authors. The authors then reach a result of 90.35 % accuracy and 0.91 F1 score. Another work [5] uses Multi-Model Deep Learning tools to accomplish three very important tasks when it comes to analyzing written documents: (i) translation, (ii) summarization, and (iii) document classification. The authors' approach, creating a multitasking model, allowed better performance in all three tasks compared to the performance obtained by isolated models.

3 Objective and practical importance of this work

The objective of this paper is to develop a model for the classification of legal proceedings in three possible classes of status: (i) archived proceedings, (ii) active proceedings and (iii) suspended proceedings. Each proceeding is made up of a sequence of short texts written by the courts that we will call "motions", which relate to the current state of proceedings, but not necessarily to their status. The three possible classes are given in a certain instant in time, and are decided by the courts, which may be temporary or permanent. In addition to focusing on the construction of a good classifier, we will also value the interpretability of the results achieved, given the importance of understanding the decisions made by models in the legal area.

These criteria have been chosen because they are a key feature to any task related to legal proceedings in Brazil. Although there are 90 different Courts in Brazil (State, Labour, Federal and others) – plus the Supreme Court –, all legal proceedings in Brazil must be included in one of the three presented classes (Archived, Active, Suspended). According to the National Council of Justice (CNJ) report[2], in the end of 2018 there were 64.6 million active proceedings and 14.1 were suspended in that year. In the same period of 2018, 31.9 million legal proceedings were definitively closed, that is, archived.

The three labels of interest (Archived, Active, Suspended) reflect the most practical classifications of the status of proceedings. Although in Procedural Law there may be some other subtle categories of analog status for proceedings (such as extinction and dismissal), the status of being archived, active or suspended is related to the activities of all personnel involved with these proceedings. For example, the suspension of a proceeding means that, even if not extinct

[2]The report can be found here `https://www.cnj.jus.br/wp-content/uploads/conteudo/arquivo/2019/08/justica_em_numeros20190919.pdf`

(and therefore subject to reactivation of the same lawsuit), and from a practical view these proceedings are out of the judiciary routine of certain portfolios from courts, law firms, civil associations or legal aid organizations.

In spite of the status of a proceeding being an objective information, sometimes it can be hard for public or private organizations to track it due to the size of their portfolios and because the information are mainly non-structured and can be spread in hundreds of separate individual Courts' web pages. It must also be noted that approximately half of the proceedings in Brazil have a small number of big players as plaintiffs of defendants, as usual in a contemporary mass litigation society. Thus, our work may help big public and private organizations to better handle their portfolios and will add value to Brazilian society as a whole.

4 Data

Our data is composed by two datasets: a dataset of $3 \cdot 10^6$ unlabelled motions and a dataset containing 6449 legal proceedings, each with an individual and variable number of motions, but which have been labeled by law experts. These datasets are random samples from the first (São Paulo) ans third (Rio de Janeiro) biggest State Courts. State Courts handle the most variable types of cases throughout the Courts in Brazil, and are responsible for 80% of the total amount of lawsuits. Therefore, these datasets are representative of a very significative portion of the variable use of language and expressions in Courts vocabulary. Since classifying sets of texts is a complex task and our dataset of labeled proceedings is not very large, we used the unlabelled texts dataset for the embedding learning of words and expressions in the legal context and we used the second dataset to create a model for the legal proceedings classification. The distribution of the legal proceedings' labels can be seen in Table 1:

	%	N
Archived (class 1)	47.14%	3040
Active (class 2)	45.23%	2917
Suspended (class 3)	7.63%	492
Total	100%	6449

Table 1: Distribution of legal proceedings' labels

5 Methodology

5.1 Text preprocessing

The first step before applying any Natural Language Processing or Machine Learning model to text is to preprocess the raw data obtained in text form. This step is crucial to the success of the application, as we make the analysis more computer friendly, avoiding, among other things, over-parameterization of the models used, which can undermine the their performance. We applied the three points below, which are standard in the literature of NLP:

- *Uppercase to lowercase conversion*: uppercase and lowercase strings are understood as different things by the computer when applying the models. Often, having or not capitalized words in your body does not change the meaning of the text, as is the case at the beginning of sentences. In order to avoid the problem of model over-parameterization, we will standardize the words in the body of the text by converting the uppercase characters to lowercase;
- *Stop words removal*: in many cases, some words add little information to the texts. We will evaluate which words will be removed without much loss of information in order to avoid the problem of over-parameterization. Words like "a" and "of" are some examples;
- *Noise removal and standardization of expressions*: noise removal is the removal of undesirable elements that may be intrinsic to the raw texts or arise by obtaining the data from the court's website. Examples are the conversion of the terms "state law" and "federal law" to "law" and the removal of punctuation and other undesired symbols;

5.2 Embedding learning

The construction of words and expressions embeddings in this work is completely unsupervised, given the small number of labeled text sequences - we then used a mass of $3 \cdot 10^6$ motion texts, all from unlabelled proceedings. Once we have

the mass of preprocessed texts, the next step is to tokenize them - in this step we use the method proposed in [6] in order to identify presence words that generally appear together and which should be considered as unique tokens[3]. Applying this methodology twice in sequence, with threshold=1, we can identify which sets of 2 to 4 words should be considered as unique tokens. After the tokenization of the texts, we then use the model specified in [7] (size=100, window=5)[4] and extract the vector representations for each of the tokens in the vocabulary - this model is a version of Word2Vec Continuous Bag of Words model, where a vector is also added for each of the documents. After obtaining each of the vector representations of terms and expressions, we normalize them to have a unitary euclidean norm, which will facilitate the interpretability of the classification model as we will show in Section 5.5.

5.3 Representation of texts in matrix form

Before describing the neural network used in the classification task, we need to understand what each text will look like after learning the embedded representations for the tokens in our vocabulary. First of all, it is important to remember that each legal proceeding we want to classify is consisted by a sequence of texts of varying length, called motions. We are now interested in knowing the format of each of the texts in question. Each motion will be represented by an array of dimensions $R \times D$ where R is the maximum number of tokens allowed for each of the texts and D the size of the embeddings - in our case $D = 100$. One can see the distribution function of the number of tokens per motion in Figure 1:

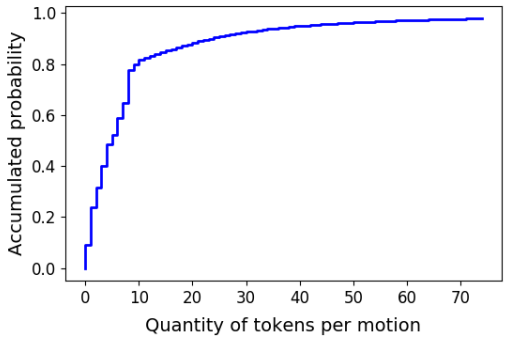

Figure 1: Distribution Function of the number of tokens per text

We have noticed that over 90% of the motions have a maximum of 30 tokens, so we decided to set a ceiling of $R = 30$ tokens, selecting the first tokens, and completing texts that have less than R tokens using null vectors of dimension D (*zero-padding*). One can see in Figure 2 how we converted each of the texts to matrix form:

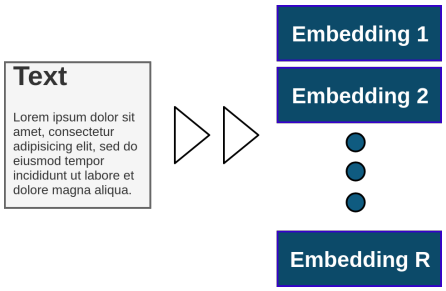

Figure 2: Representation of texts in matrix form

[3]This method is one of the versions of "Phrase" function that is implemented in the Gensim package `https://radimrehurek.com/gensim/`.

[4]We tested many configurations, e.g. windows=5, 10, 15 and size=50, 100, 150, and we chose to work with the more parsimonious and most performing one, according to the classification results.

5.4 Construction of the Neural Network for legal proceeding classification

Now that we know the format of the representation of each text that will use to feed our neural network, we can better explain how the classification model that combines convolutional filters [8] with a recurrent neural network (RNN) with Long Short-Term Memory units (LSTM) [9]) works. We mentioned that each legal proceeding is composed of a sequence of motions/texts and, as in the case of tokens, we needed to impose a ceiling on the number of motions/texts we would use. With a battery of tests, we realized that the ideal number of texts to consider is five (5), that is, we separated the last five (5) motions/texts from each of the legal proceedings and put them in chronological order - those proceedings that had less than 5 motions available were completed by zero-padding matrices, always putting the most current motions closer to the output layer, which is a Softmax function. It is possible to see in Figure 3 that generally the legal proceedings are composed of a much larger quantities of motions/texts, but for our purpose they are negligible:

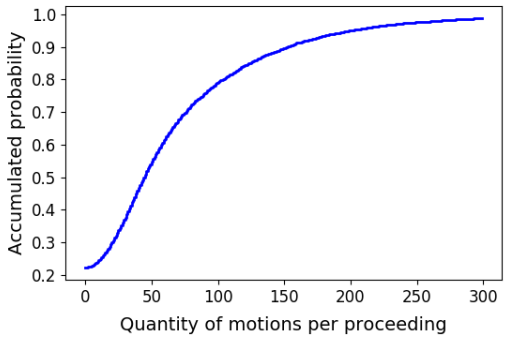

Figure 3: Distribution Function of the number of motions per legal proceeding

To extract features from each motion we used a convolutional layer with K[5] one-dimensional filters that run through each text. After extracting the features, they pass through a ReLU activation function and then are selected according to the *max-over-time pooling* procedure proposed in [10], that is, we kept only one feature per filter, the one with the highest value - each motion/text will be represented by only K numbers[6], that feed the Recurrent Neural Network with LSTM units with H[7] hidden neurons. After processing the data using the RNN, the legal proceeding is then classified by a Softmax function. In order to give an interpretable appeal to the solution, in the learning process[8] of the neural network, we constrain the euclidean norm of filters to be equal to one. Later in Section 5.5 and in Section 6 we will show that we can easily compare filters learned by the network with the embeddings representations of tokens present in our vocabulary. There is an illustration of the neural network used in Figure 4.

[5] It will be determined through a cross validation procedure.

[6] Thus, each legal proceeding is represented by $5K$ numbers (5 motions and K features per motion)

[7] Also determined through a cross validation procedure.

[8] We used the backpropagation algorithm combined with Adam Optimizer. Also, we used 200 epochs and batch size equals 500.

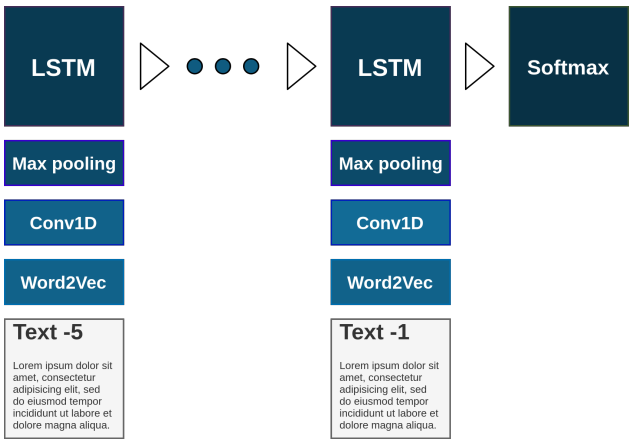

Figure 4: Neural Network Architecture

5.4.1 Describing how our neural network works with more details

Now that we know how the network works in classifying legal proceedings, it's important that we present those ideas in mathematical language, which will help us make things clearer. Let (i) i be the index of a legal proceeding [9], (ii) $t \in \{-5, \dots, -1\}$ a index for a text/motion of i proceeding, where -1 denotes the most current text and -5 the least current ttext taken into account, (iii) $n \in [30]$ an index [10] of embedded tokens in the text t from proceeding i and (iv) $\mathbf{f}_k \in \mathbb{R}^{100}$ is the vector representing the k-th convolutional filter, $k \in [K]$. We then define the following quantity z_{itnk}, which is the feature extracted by the filter $\mathbf{f}_k$ from tokem $\mathbf{x}_{itn} \in \mathbb{R}^{100}$, that is, n-th tokem from t-th motion/text from i-th proceeding:

$$z_{itnk} = \text{ReLU}(\mathbf{x}_{itn} \cdot \mathbf{f}_k) \tag{1}$$

Note that we removed the constant neuron, which represents the bias. Furthermore, the final feature extracted by the $\mathbf{f}_k$ filter from the t-th motion/text from i-th proceeding right after applying *max-over-time pooling* procedure is given by the quantity z^*_{itk} as follows:

$$z^*_{itk} = \max\left\{z_{itnk}\right\}_{n=1}^{30} \tag{2}$$

Grouping those quantities through index k in an array, we have the following vector that we will use to feed our recurrent neural network with LSTM units:

$$\mathbf{z}^*_{i,t} = (z^*_{it1}, \dots, z^*_{itK}) \tag{3}$$

The probability vector of i-th legal proceeding belonging to one of the three possible classes/status, $\mathbf{p}_i$, is given by the function $\boldsymbol{h}$ which is a recurrent neural network (RNN/LSTM) with a time depth of 5:

$$\mathbf{p}_i = \boldsymbol{h}(\mathbf{z}^*_{i,-1}, \dots, \mathbf{z}^*_{i,-5}) \tag{4}$$

Given $\mathbf{z}^*_{i,-1}$ refers to the most current network input and $\mathbf{z}^*_{i,-5}$ refers to the least current input. For a class $j \in [3]$, we can also write the individual predicted probability as $p_{ij} = h_j(\mathbf{z}^*_{i,-1}, \dots, \mathbf{z}^*_{i,-5})$. It's not explicit, but this time, as well as all the others not mentioned, we included the constant neuron to take the bias into account.

[9] i can represent an out of sample proceeding.
[10] Consider [N] = $\{1, \dots, N\}$.

5.5 Interpretability

5.5.1 What are the filters looking for?

In the process of feature extraction performed by the convolutional layer of the network, we have that each of the 5 filters go through all 30 embedding representations of tokens present in each text performing scalar products, which in themselves are already similarity metrics. As we discussed earlier, each of the embeddings representations and filters were constrained to have unitary euclidean norm and that means the scalar product between the filters and embeddings representations will give us the value of the cosine of the shortest angle formed between the vectors, i.e. the cosine similarity between they. Mathematically, we have:

$$\mathbf{x}_{itn} \cdot \mathbf{f}_k = \|\mathbf{x}_{itn}\| \, \|\mathbf{f}_k\| \cos(\theta_{itnk}) \tag{5}$$
$$= \cos(\theta_{itnk}) \tag{6}$$

If θ_{itnk} is the shortest angle formed between the vectors $\mathbf{x}_{itn}$ and $\mathbf{f}_k$. Thus, we can rewrite z_{itnk} as below:

$$z_{itnk} = \text{ReLU}\big[\cos(\theta_{itnk})\big] \tag{7}$$

Which equals to $\cos(\theta_{itnk})$ if $\theta_{itnk} \in [0, \pi/2]$. In the learning process, the network learns representations for filters that tend to minimize the loss (cross entropy) function when classifying. By constraining the vectors to have unitary euclidean norms, while learning the best weights for the convolutional layer, the network aligns[11] the filters representations to those representations of the tokens that help the most in the task of classifying legal proceedings. Then, analyzing the final representations of the filters, we can have insights on the patterns that the network looks for in the texts. In order to better understand what those patterns are, we are going to take a look at the tokens which have the closest representations to the filters according to cosine similarity.

5.5.2 How do features extracted by each filter relate to classification?

To interpret how each filter relates to the classification task, we will use the Partial Dependence Plots [12]. To explain the concept, we will first introduce a new notation. If Y_i is the random variable that denotes the class of the i-th proceeding, then we can rewrite p_{ij} as follows:

$$p_{ij} = \hat{\mathbb{P}}\big(Y_i = j \;\big|\; \mathbf{z}^*_{i,-1}, \dots, \mathbf{z}^*_{i,-5}\big) \tag{8}$$
$$= \hat{\mathbb{P}}\big(Y_i = j \;\big|\; z^*_{i,-1,1}, \dots, z^*_{i,-5,K}\big) \tag{9}$$

Moreover, in order to help us define the partial dependence function, we will write $\mathbf{z}^*_i = (\mathbf{z}^*_{i,-1}, \dots, \mathbf{z}^*_{i,-5})$ as the concatenation of the vectors. When we want to talk about the features themselves and not their instances in the i individual, we can rewrite z^*_{itk} as Z^*_{tk}, $\mathbf{z}^*_{i,-1}$ as $\mathbf{Z}^*_{-1}$ and $\mathbf{z}^*_i$ as $\mathbf{Z}^*$. Given all these notations, the partial dependence function on Z^*_{tk} feature predicting j class probability, with $t = -1$ and $k = 1$ for example, is given by:

$$g_{j,Z^*_{-1,1}}(z) = \mathbb{E}_{\mathbf{Z}^* \setminus Z^*_{-1,1}}\Big[\mathbb{P}\big(Y = j \;\big|\; z, Z^*_{-1,2}, \dots, Z^*_{-5,K}\big)\Big] \tag{10}$$

With $\mathbf{Z} \setminus Z^*_{-1,1}$ denoting the $\mathbf{Z}^*$ vector without the first original feature. Here we work with the $z^*_{-1,1}$ feature for pure practicality, but the definition is valid for any of the features. The empirical version of the partial dependence function for the same feature is given by the following:

$$\hat{g}_{j,Z^*_{-1,1}}(z) = \frac{1}{m} \sum_{i=1}^{m} \hat{\mathbb{P}}\big(Y_i = j \;\big|\; z, z^*_{i,-1,2}, \dots, z^*_{i,-5,K}\big) \tag{11}$$

In this paper, we will calculate this function according to the test set data and center it on zero, so that it is easier to make comparisons between plots - so we will be interested in variations in the average probabilities of the j class given variations in an specific feature.

[11] By 'aligning' we mean 'approximating' according to the cosine similarity metric.

[12] See [11] for a more detailed explanation.

5.6 Hyperparameter tuning

Hyperparameters are settings used to control the behaviour of learning algorithms which are not adapted by the algorithm itself [12]. We have chosen to keep some of the hyperparameters fixed and to tune the rest of them in a simple cross-validation procedure using the grid search approach. Table 2 shows a summary about the values tested or fixed for the hyperparameters that we worried about while building the model.

Hyperparameter	Fixed/Tuned	Values tested	Fixed value
Optimizer	Fixed	-	Adam
Beta 1 (Adam)	Fixed	-	0.9
Beta 2 (Adam)	Fixed	-	0.999
Learning rate	Fixed	-	0.001
# Epochs	Fixed	-	200
Batch size	Fixed	-	500
# Convolutional filters	Tuned	1, 2, 3, 4, 6, 8, 10	-
# LSTM hidden state size	Tuned	10, 20, 30, 40, 50, 75, 100	-
LSTM weights regularization (l2)	Tuned	0.0, 0.001, 0.01, 0.1	-

Table 2: Hyperparameters for the classification model

5.7 Training, validation and test sets

In order to train our classifier, we randomly splitted our labeled dataset in three parts as shown in the Table 3:

	%	N
Training set	70.0%	4514
Validation set	10.0%	645
Test set	20.0%	1290
Total	100%	6449

Table 3: Training, validation and test sets

We used the training set to fit the model, the validation set to choose the best hyperparameters and the test set just to check the performance of the final model.

6 Results

6.1 Hyperparameters

For the choice of hyperparameters, we separated the 25 best combinations of values for the hyperparameters[13] according to the accuracy obtained by the model in the validation set. The top 25 combinations showed no differences in accuracy that were statistically significant at a significance level of 5%. Therefore, we opted for the combination that made the final model as simple as possible. In Table 4 one can check the final values for the hyperparameters for the classification model:

[13] Out of 196 possible combinations.

Hyperparameter	Value
Optimizer	Adam
Beta 1 (Adam)	0.9
Beta 2 (Adam)	0.999
Learning rate	0.001
# Epochs	200
Batch size	500
# Convolutional filters	3
# LSTM hidden state size	20
LSTM weights regularization (l2)	0.001

Table 4: Final hyperparameters for the classification model

6.2 Proceeding classification task performance

The classifications were made by selecting the most likely class, given the features, according to the estimated model. A first point that is interesting to analyze is the behavior of learning curves, which denote accuracy, during the training phase. One can check that behavior in Figure 5.

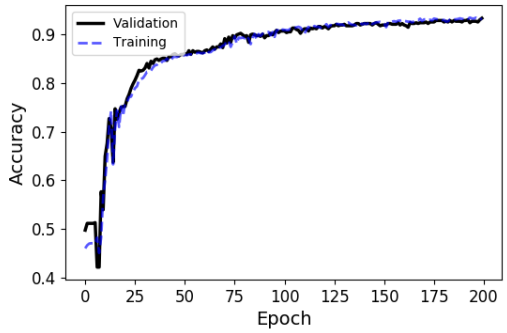

Figure 5: Accuracy

In the process of training our final model, it was not possible to verify accentuated overfitting, due to the parsimonious architecture of the network, counting with only 2283 trainable parameters. Now it is necessary to evaluate the performance in the test set. The most straightforward way to get an overview of classifier performance in this case is by using the confusion matrix, which compares true labels with those predicted by the model. Here our classes are: archived proceedings (1), active proceedings (2) and suspended proceedings (3). It is possible to check in Figure 6 the joint distribution that characterizes the confusion matrix:

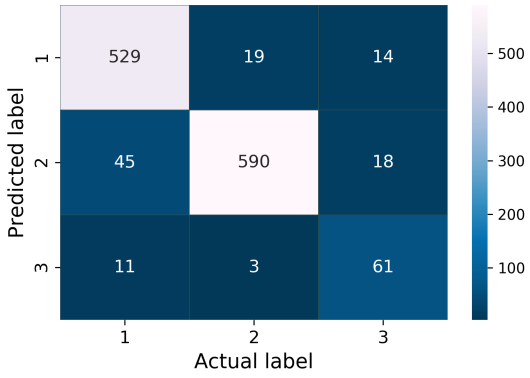

Figure 6: Confusion matrix in the test set

It is possible to notice a great performance of the model with accuracy of 0.91 ± 0.02. Let's now look at some other metrics for each class in Table 5[14].

Class	F1 Score	Precision	Recall	N
Archived (class 1)	0.92 ±0.02	0.94 ±0.02	0.9 ±0.02	585
Active (class 2)	0.93 ±0.01	0.9 ±0.02	0.96 ±0.01	612
Suspended (class 3)	0.73 ±0.07	0.81 ±0.09	0.66 ±0.1	93

Table 5: Evaluation metrics by class

For the first two classes, we have great classifier performance across all evaluated metrics. It can be seen that by classifying the examples belonging to the minority class, which is the suspended proceeding class, the model performed much less than the other cases when we look at the F1 Score and Recall metrics. This is probably due to the fact that this is a less well-defined class than the other two, since processes that are suspended may still be archived or become active. In addition to looking at the performance metrics for each class separately, we can look at the Table 6 which contains summary measures that characterize the model as a whole, which are the simple (macro-averaging) and weighted (micro-averaging) averages of the metrics already seen[15].

Average	F1 Score	Precision	Recall	N
Simple average	0.86 ±0.02	0.89 ±0.03	0.84 ±0.03	1290
Weighted average	0.91 ±0.01	0.91 ±0.02	0.91 ±0.02	1290

Table 6: Aggregate analysis of evaluation metrics

Overall, we can consider that the model performed considerably well by looking at both simple and weighted averages.

6.3 Interpretability of results

6.3.1 What are the filters looking for?

In order to better understand what are the patterns extracted by the convolutional layer of the neural network, let's look at the embeddings representations of tokens in our vocabulary which have the closest representations to the filters according to cosine similarity. In Table 7 we can see which tokens[16] most closely resemble our filters after they are learned:

[14]The 0.95 confidence intervals were calculated using a bootstrap procedure.

[15]The 0.95 confidence intervals were calculated using a bootstrap procedure.

[16]Tokens were translated from portuguese to english.

	Tokens	Cosine similarity
Filter 1	*respondent party*	0.52
	clarification appeal entered	0.49
	granted	0.49
	claimant party	0.48
	declared opinion	0.48
Filter 2	*final remittance to origin*	0.54
	final storage of docket	0.53
	payment through court fees invoice	0.44
	apt determine	0.39
	leased vehicle	0.39
Filter 3	*final storage of docket*	0.52
	archived	0.5
	temporarily storaged docket	0.48
	non-reactivated proceeding	0.48
	central storage	0.47

Table 7: Similarity between filters and their most similar tokens

We can see that the patterns sought by the neural network do have to do with the classifications we want to make. For example, the expressions 'final storage of docket' and 'final remittance to origin' indicate archiving of proceedings (class 1) and the expression 'temporarily storaged docket' may indicate suspension (class 3).

6.3.2 How do features extracted by each filter relate to classification?

Now that we have a notion about the patterns sought by the neural network in texts/motions, let's try to better understand how each of those patterns relates to the classification of legal processes into the three possible classes of status, which are (1) archived proceedings, (2) active proceedings and (3) suspended proceedings. Regarding to Figure 7, it is possible to notice that the partial dependence functions are decreasing in all plots but the one related to the active proceedings. This is understandable because those expressions in Table 7 which are related to filter 1 are more common to appear when a proceeding is active, e.g. 'granted' refers to a judgement made in a Court.

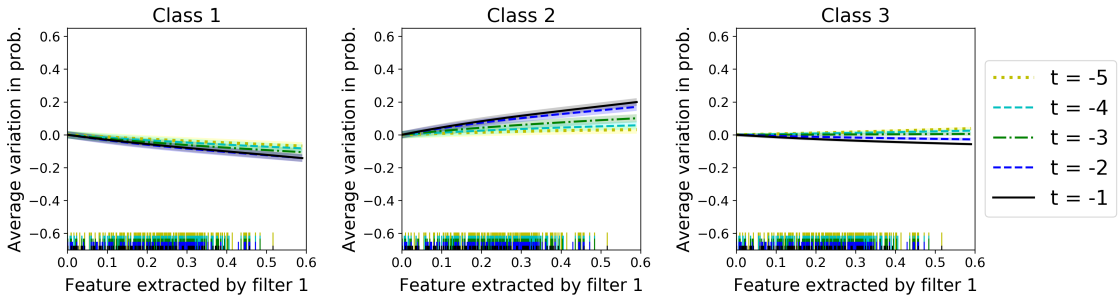

Figure 7: Partial dependence plots: varying features extracted by filter 1

The patterns extracted by filter 2, in Figure 8, explain which legal proceedings are likely to be archived but not suspended or active. These facts contrast with what will be seen in Figure 9, since filter 3 looks for patterns that favor the classes archived and suspended in relation to active, e.g. 'final storage of docket', 'archived' and 'temporarily storaged docket' are expression which may indicate whether a proceeding is archived or suspended.

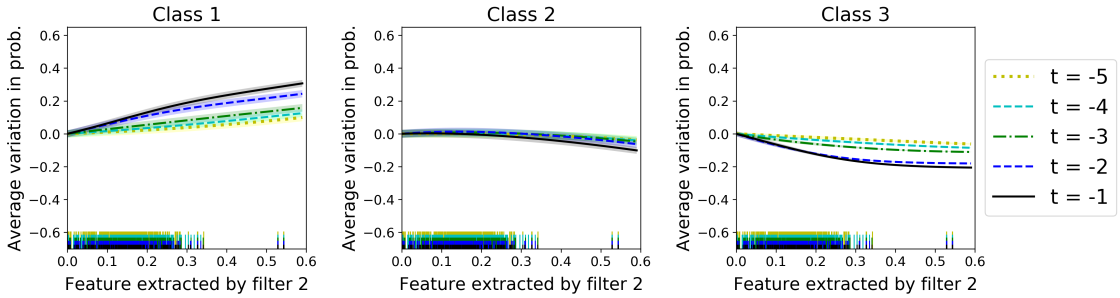

Figure 8: Partial dependence plots: varying features extracted by filter 2

Patterns extracted by filter 3, present in Figure 9, have high explanatory power in predicting which processes are likely to be archived or suspended but not be active. Another interesting point to pay attention to is that more current information is, greater its importance - in the first and second plots, there is a large gap between the importance of the most current text/motion and others even though the second most current text may still have some importance. Regarding the suspended/third class, one can realize that the most recent information is as important as the second most recent information, which is an interesting fact.

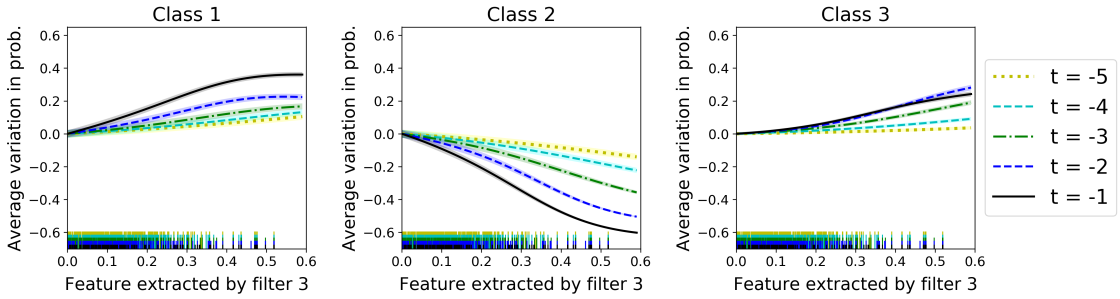

Figure 9: Partial dependence plots: varying features extracted by filter 3

To conclude this section, we would like to highlight two points that we find most interesting regarding these results: (i) the results were very intuitive regarding the link between patterns search by the network in the texts and the output of the classification model and (ii) it is possible to notice that more recent information tends to have greater importance in the decision of the neural network, which makes sense in the legal context.

7 Conclusion

This work aimed to develop a model for the classification of legal processes composed of sequential texts. During the development of the model, we wanted to have a model that performed very well on the classification task, had a parsimonious architecture and that we could gain insight into how decisions are made. We believe that the major contribution of this work is precisely the way we solve an important problem, which is classifying legal proceedings' status, combining several types of techniques to analyze sequences of texts in chronological order, which are so common in the legal context. The results obtained were satisfactory both in terms of classification and interpretability, which also brings importance to this work.

References

[1] Harry Surden. Machine learning and law. *Wash. L. Rev.*, 89:87, 2014.

[2] Nikolaos Aletras, Dimitrios Tsarapatsanis, Daniel Preoţiuc-Pietro, and Vasileios Lampos. Predicting judicial decisions of the european court of human rights: A natural language processing perspective. *PeerJ Computer Science*, 2:e93, 2016.

[3] Octavia-Maria Sulea, Marcos Zampieri, Mihaela Vela, and Josef van Genabith. Predicting the law area and decisions of french supreme court cases. *arXiv preprint arXiv:1708.01681*, 2017.

[4] N Correia da Silva, FA Braz, DB Gusmão, FB Chaves, DB Mendes, DA Bezerra, GG Ziegler, LH Horinouchi, MHP Ferreira, PHG Inazawam, et al. Document type classification for brazil's supreme court using a convolutional neural network. 2018.

[5] Ahmed Elnaggar, Christoph Gebendorfer, Ingo Glaser, and Florian Matthes. Multi-task deep learning for legal document translation, summarization and multi-label classification. *arXiv preprint arXiv:1810.07513*, 2018.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[7] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[8] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[11] Christoph Molnar. *Interpretable Machine Learning*. 2019. `https://christophm.github.io/interpretable-ml-book/`.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.