PREDICTING LEGAL PROCEEDINGS STATUS: AN APPROACH BASED ON SEQUENTIAL TEXTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine learning applications in the legal field are numerous and diverse. In order to make contributions 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 brazilian 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. Our work may help big public and private organizations to better handle their portfolios and will add value to Brazilian society as a whole.[1]

1 INTRODUCTION

In this work we make extensive use of natural language processing (NLP) and machine learning tools to classify legal proceedings. Although there are some efforts to apply machine learning and NLP 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. For example, Aletras et al. (2016) 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%. da Silva et al. (2018), a recent Brazilian study, makes use of Convolutional Neural Networks to classify documents analyzed by the Brazilian Supreme Court (STF), achieving significant results. The authors then reached a result of 90.35 % accuracy and 0.91 F1 score.

2 OBJECTIVE

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, which may be temporary or permanent, and are decided by the courts. 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 - our work may help big public and private organizations to better handle their portfolios and will add value to Brazilian society as a whole.

[1]The code (Jupyter Notebooks) used in this work as well as the datasets can be found in `https://bit.ly/36yJZY3`. The data can also be found in `https://doi.org/10.6084/m9.figshare.11750061.v1`.

3 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) and 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 legal proceedings' can be splitted in: (i) Archived (class 1) 47.14% of total, with 3040 proceedings, (ii) Active (class 2) 45.23% of total, with 2917 proceedings and (iii) Suspended (class 2) 7.63% of total, with 492 proceedings.

As soon as we got the text data, we had to preprocess it. 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 any application, as we make the analysis more computer friendly, avoiding, among other things, over-parameterization of the models used, which can undermine their performance. We applied the three points below, which are standard in the literature of NLP: (i) uppercase to lowercase conversion, (ii) stop words removal and (iii) removal of undesirable elements, such as punctuation.

4 METHODOLOGY

4.1 EMBEDDING LEARNING AND REPRESENTATION OF TEXTS

We used a mass of $3 \cdot 10^6$ motion texts, all from unlabelled proceedings, to learn vectors representations for word and expressions. We used the method proposed by Mikolov et al. (2013b) in order to identify which sets of 2 to 4 words that generally appear together and which should be considered as unique tokens. After tokenization, we then use the model specified in Mikolov et al. (2013a) (CBOW Word2Vec) (size=100, window=5)[2] and extract the vector representations for each of the tokens in the vocabulary. Then we normalized the learned representations to have a unitary euclidean norm - this is important as we will see in Section 4.3. After all, we represent each motion/text by a matrix 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 $D = 100$. 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.

4.2 CONSTRUCTION OF THE NEURAL NETWORK FOR LEGAL PROCEEDING CLASSIFICATION

Our legal experience is that the last 5 motions contain enough information for our purpose. Then, we separated the last five (5) motions/texts from each of the legal proceedings and put them in chronological order, using zero-padding when necessary. To extract features from each motion we used a convolutional layer (Kim, 2014) with K 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 by Collobert et al. (2011), 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[3], that feed the Recurrent Neural Network (RNN) with Long Short-Term Memory LSTM units (Hochreiter & Schmidhuber, 1997) with hidden state size H. 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, as we show ahead, we constrained the euclidean norm of filters to be equal one. There is an illustration of the neural network used in Figure 1.

[2]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.

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

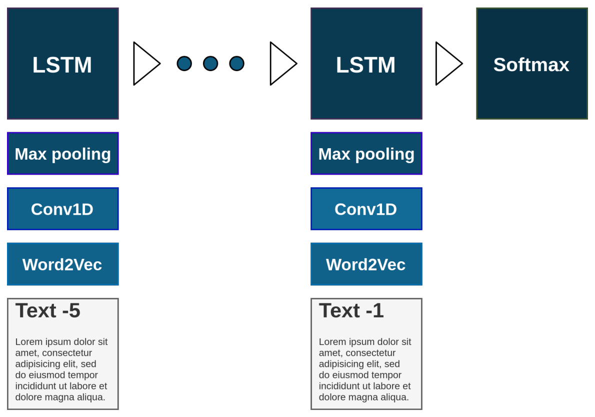
LSTM
LSTM
Softmax
Max pooling
Max pooling
Conv1D
Conv1D
Word2Vec
Word2Vec
Text -5
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Text -1
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Figure 1: Neural Network Architecture

4.3 Interpretability

4.3.1 What are the filters looking for?

Now that we know how the network works in classifying legal proceedings, it's important to understand what are the features extracted by the convolutional layer. Let (i) i be the index of a legal proceeding [4], (ii) $t \in \{-5, \ldots, -1\}$ a index for a text/motion of i proceeding, where -1 denotes the most current text and -5 the least current text taken into account, (iii) $n \in [30]$ an index [5] 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[6], 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 $z^*_{itk} = \max\{z_{itnk}\}^{30}_{n=1}$. The $5K$ extracted features are processed by the recurrent neural network outputting three probabilities, one for each class. 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. Thus, if θ_{itnk} is the shortest angle formed between the vectors $\mathbf{x}_{itn}$ and $\mathbf{f}_k$ we can write $z_{itnk} = \text{ReLU}\big[\cos(\theta_{itnk})\big]$. In the learning process, the network aligns the filters representations to those representations of the tokens that help the most in the task of classifying legal proceedings. Then, after the training phase, we just have to check which are the closest tokens' representations to our filters' representations to gain some interpretability.

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

To interpret how each filter relates to the classification task, we will use Partial Dependence Plots[7]. To help us, see that when we want to talk about the features themselves, i.e. random variables and vectors, and not their instances in the i individual, we can rewrite z^*_{itk} as z^*_{tk}, for example. In this paper, we will calculate the partial dependence functions according to the test set data and center it on zero, so that it is easier to make comparisons between plots - we will be interested in average variations in the probabilities of the j class given variations in an specific feature z^*_{tk}.

[4] i can represent an out of sample proceeding.
[5] Consider [N] = $\{1, \ldots, N\}$.
[6] We kept it in all other occasions.
[7] See Molnar (2019) for a more detailed explanation.

Table 1: Aggregate analysis of evaluation metrics

	Macro averaging			Micro averaging		
Features	F1 Score	Precision	Recall	F1 Score	Precision	Recall
CNN	**0.89 ±0.02**	0.92 ±0.02	**0.87 ±0.03**	**0.93 ±0.01**	**0.93 ±0.01**	**0.93 ±0.01**
Doc2Vec	0.82 ±0.03	0.85 ±0.03	0.8 ±0.03	0.85 ±0.01	0.86 ±0.02	0.85 ±0.02
TFIDF	0.88 ±0.02	**0.93 ±0.02**	0.85 ±0.03	0.91 ±0.01	0.92 ±0.01	0.92 ±0.02

4.4 Hyperparameter tuning and dataset splitting

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. The ones fixed and their values are: (i) ”Optimizer” = ”Adam”, (ii) ”Beta 1” (Adam) = 0.9, (iii) ”Beta 2” (Adam) = 0.999, (iv) ”Learning rate” = 0.001, (v) # of Epochs = 200 and (vi) Batch size = 500. The ones to be tuned and values testes are: (i) # of Convolutional filters (K) = $(3,\ 5,\ 8,\ 12)$, (ii) LSTM hidden state size (H) = $(10,\ 30,\ 50,\ 75,\ 100)$ and (iii) LSTM weights l1 penalization strength (λ) = $(.0,\ .0001,\ .0003,\ .0005,\ .0007,\ .0009,\ .0011,\ .0013,\ .0015,\ .0016,\ .0018,\ .002,\ .0025,\ .003)$. In order to train and assess our classifier and tune hyperparameters, we splitted at random our labeled dataset in three parts: training set (70%), validation set (10%) and test set (20%). 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.

5 Results

5.1 Hyperparameters

Our criteria to choose the best combination of hyperparameters was to choose those values who gave us the higher accuracy in the validation set. Out of 280 possible combinations of values we chose the following values for the tuned hyperparameters: (i) $K = 12$, (ii) $H = 10$ and (iii) l1-$\lambda = .0001$.

5.2 Proceeding classification task performance

In order to present the results and compare them to those obtained by similar alternatives, we will consider two other ways to extract features from the texts (other than convolutional filters), maintaining the recurrent neural network part, as it is important for us to take into account the chronological order of the facts. The other two ways to extract features are applications of the Doc2Vec (Le & Mikolov, 2014) and TFIDF (Salton & McGill, 1986) models, which were trained beforehand in the unlabeled dataset. For the Doc2Vec alternative we kept the specifications for the Word2Vec model that we discussed in Section 4.1. For the TFIDF alternative, we imposed a ceiling of 2000 tokens, keeping the more frequents in the corpus. For both alternatives we applied the processing steps described in Section 3 and 4.1[8]. First, it is important to say that our main model had the greatest accuracy (0.93 ± 0.01) compared to the Doc2Vec (0.85 ± 0.02) and TFIDF (0.92 ± 0.02) alternatives. Moreover, as one can see in Table 1[9], we obtained excellent results with our main model proposal as well as the second best alternative. Despite our main proposal achieving similar results to another option, it is in its simplicity[10] and interpretability that this solution stands out, as we will see next.

5.3 Interpretability of results

5.3.1 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

[8]The hyperparameters for the alternative models were tuned as it is described in the Supplementaty Material.

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

[10]Our main model has 2,153 trainable weights while the Doc2Vec benchmark has 15,813 and the TFIDF alternative has 243,813. One can see that our main model is much simpler, then less prone to overfitting and easier/faster to train.

closest representations to the filters according to cosine similarity. As long as we have 12 filters in our model, which is a big quantity, we are going to focus in three specific filters (1, 9 and 11), which bring interesting results - the full results will be available in the Supplementary Material. Regarding the *filter 1*, we have[11]: (i) "*final storage of docket*" (0.46), (ii) "*final remittance to origin*" (0.45). Regarding the *filter 9*, we have: (i) "*emitted*" (0.47), (ii) "*certificate*" (0.43). Regarding the *filter 11*, we have: (i) "*temporarily stored docket*" (0.55), (ii) "*docket remain in clerk*" (0.5). It seems filter 1 and 11 are important for us while filter 9 search for patters not directly linked to the classification.

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

The patterns extracted by filter 1, in Figure 2, explain which legal proceedings are likely to be archived but not suspended or active, which can easily make sense when one sees those expressions linked to filter 1, e.g. 'final storage of docket' and 'final remittance to origin'. Regarding to filter 11, it is possible to notice that the partial dependence functions are decreasing in all plots but the one related to the suspended proceedings. This is understandable because the expressions linked to filter 11, as seen in Section 5.3.1, are more common to appear when a proceeding is suspended, e.g. 'temporarily stored docket'. On the other hand, patterns extracted by filter 9, presented in Figure 2, have almost no impact in the decision of the neural network as expected. Also, it seems that more recent information is more important. Overall, the results are very intuitive.

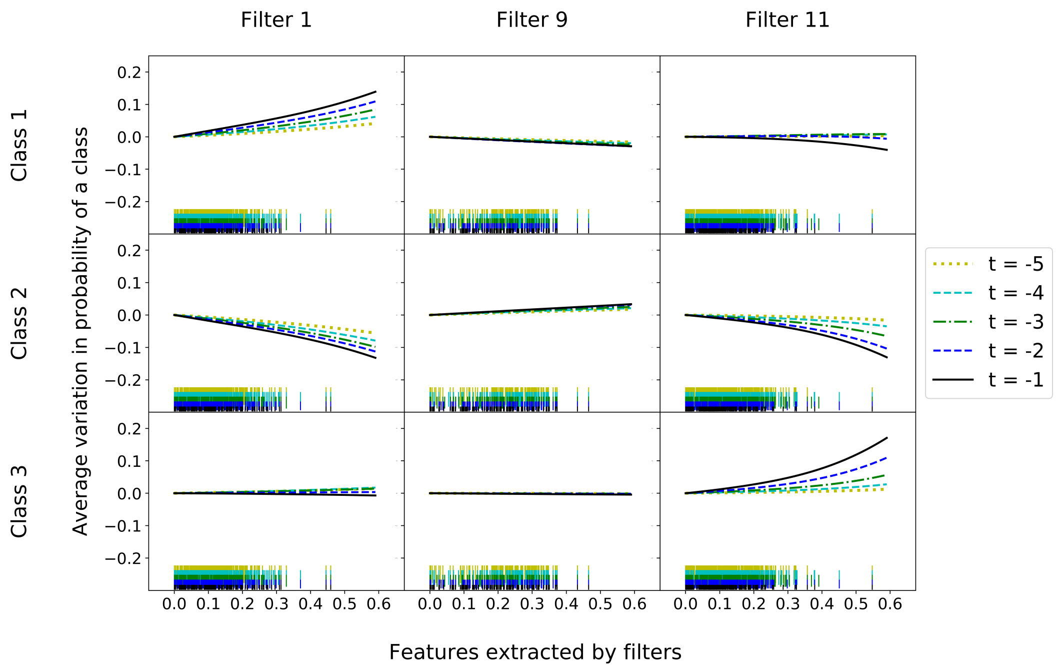

Figure 2: Partial dependence plots

6 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.

[11]Cosine similarity in parentheses.

References

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.

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.

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.

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

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

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

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*, pp. 3111–3119, 2013b.

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

Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.