

# Predicting Legal Proceedings Status: An approach Based on Sequential Texts

Felipe Maia Polo<sup>1</sup>  
Itamar Ciochetti<sup>2</sup>  
Emerson Bertolo<sup>2</sup>

<sup>1</sup>University of São Paulo, Brazil

<sup>2</sup>Tikal Tech, Brazil

Practical ML for Developing Countries Workshop @ ICLR 2020



# Introduction

Presentation based on:

<https://arxiv.org/abs/2003.11561>

# Introduction

- Machine learning applications in the legal field are numerous and diverse;
- Examples of applications: reviewing documents, making text-based classifications or anticipating legal outcomes;
- Natural Language Processing (NLP) tools became mainstream in the legal field due to the amount of non-structured (mainly texts) data available;

# Introduction

- One important task is to classify legal proceedings according to their status:
  - ① Archived proceedings;
  - ② Active proceedings;
  - ③ Suspended proceedings;
- Although there are 90 different Courts in Brazil, all legal proceedings in Brazil must be included in one of the three presented classes;
- Developing an algorithm to accomplish that task could help big public/private organizations to manage their portfolios;

# Objective

- Develop an algorithm to classify legal proceedings in three possible classes of status:
  - ① Archived (class 1 - 47.14% of sample);
  - ② Active (class 2 - 45.23% of sample);
  - ③ Suspended (class 3 - 7.63% of sample);
- The three possible classes are given in a certain instant in time, which may be temporary or permanent, and are decided by the courts;
- Moreover, we will also value the *interpretability* of the results, given the importance of understanding the decisions made by models in the legal area;

# Data

- 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 texts follow a chronological order;
- We use two datasets:
  - 1 A dataset of  $3 \cdot 10^6$  unlabelled motions;
  - 2 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 samples from the first and third biggest State Courts, i.e. São Paulo and Rio de Janeiro;

## Text preprocessing

- **Uppercase to lowercase conversion:** avoids the problem of over-parameterization;
- **Stop words removal:** removes words that bring little or no information;
- **Noise removal and standardization of expressions:** removes undesirable elements and standardizes expressions - e.g. "state law" and "federal law" become "law";
- **Tokenization:** we used the method proposed by [MSC<sup>+</sup>13] in order to identify which sets of 2 to 4 words that generally appear together and which should be considered as unique tokens;

## Embedding training

- We used the unlabeled dataset to train our vector representation of tokens;
- The algorithm used for that task was the Word2Vec Continuous Bag-of-Words [MSC<sup>+</sup>13] with size=100 and window=5;
- We normalized the vector representations to have a unitary euclidean norm, which facilitated the interpretability of the classification model as we will soon;



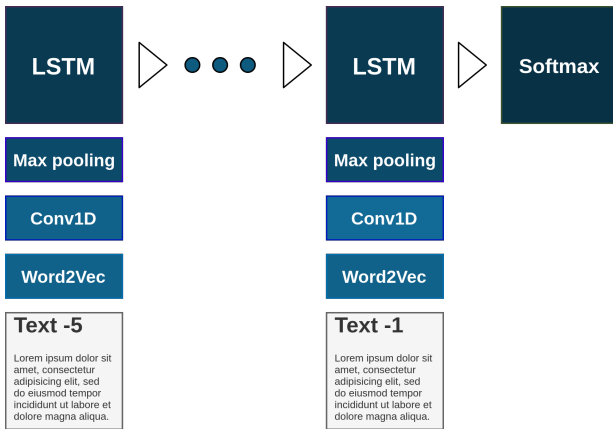
## Legal proceeding (document) representation

- Each document in our work is represented by its last 5 texts (motions);
- Each text (motion) is represented by a real matrix of dimensions  $R \times D$ , where  $R$  is the maximum number of tokens allowed for each of text 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 and using zero-padding when necessary;

# Neural Network Architecture

- We developed a classification model that combines a Recurrent Neural Network (RNN) with Long Short-Term Memory units (LSTM) [HS97] with convolutional filters [Kim14];
- The convolutional filters were used to extract the features from each text and the RNN was used to take into account the chronological order of facts;
- Using the *max-over-time pooling* procedure proposed in [CWB<sup>+</sup>11], we kept only one feature per convolutional filter - the one with the highest value. After convolution, each text is then represented by a  $K \times 1$  matrix,  $K$  being the number of filters;

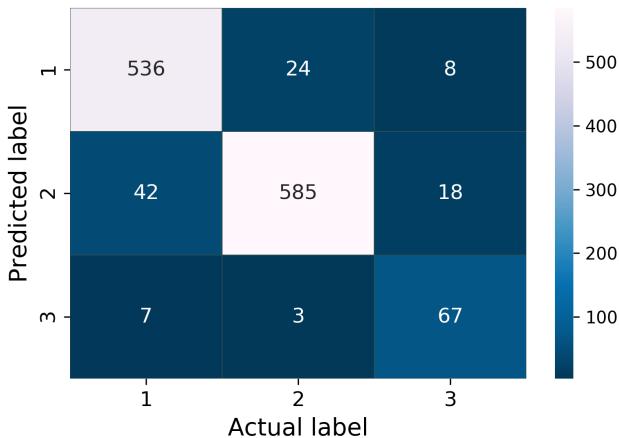
# Neural Network Architecture



# Hyperparameters tuning

Hyperparameter	Values tested/fixed
Optimizer	<b>Adam</b>
Beta 1 (Adam)	<b>0.9</b>
Beta 2 (Adam)	<b>0.999</b>
Learning rate	<b>0.005</b>
# Epochs	<b>50</b>
Batch size	<b>500</b>
# Convolutional filters (K)	3, 5, 8, <b>12</b>
LSTM hidden state size (H)	<b>10</b> , 30, 50, 75, 100
LSTM weights l1 penalization strength ( $\lambda$ )	.0, <b>.0001</b> , .0003, .0005, .0007, .0009, .0011, .0013, .0015, .0016, .0018, .002, .0025, .003

## Confusion Matrix in the test set



# Aggregate analysis of evaluation metrics

Macro averaging			
Feature extraction	F1 Score	Precision	Recall
CNN	<b>0.89 ±0.02</b>	0.92 ±0.02	<b>0.87 ±0.03</b>
Doc2Vec	0.82 ±0.03	0.85 ±0.03	0.8 ±0.03
TFIDF	0.88 ±0.02	<b>0.93 ±0.02</b>	0.85 ±0.03

Micro averaging			
Feature extraction	F1 Score	Precision	Recall
CNN	<b>0.93 ±0.01</b>	<b>0.93 ±0.01</b>	<b>0.93 ±0.01</b>
Doc2Vec	0.85 ±0.01	0.86 ±0.02	0.85 ±0.02
TFIDF	0.91 ±0.01	0.92 ±0.01	0.92 ±0.02

## Quantity of learnable weights by approach

Feature extraction	# Learnable weights
CNN	<b>2,153</b>
Doc2Vec	15,813
TFIDF	243,813

# Features extracted by convolutional filters

- Consider  $\mathbf{f} \in \mathbb{R}^{100}$  to be a convolutional filter and  $\mathbf{x} \in \mathbb{R}^{100}$  a vector representation for a specific token;
- If we constrain the norms of tokens and filters to be unitary, that is  $\|\mathbf{f}\| = \|\mathbf{x}\| = 1$ , then the convolution is a dot product and returns:

$$\mathbf{f} \cdot \mathbf{x} = \cos(\theta) \quad (1)$$

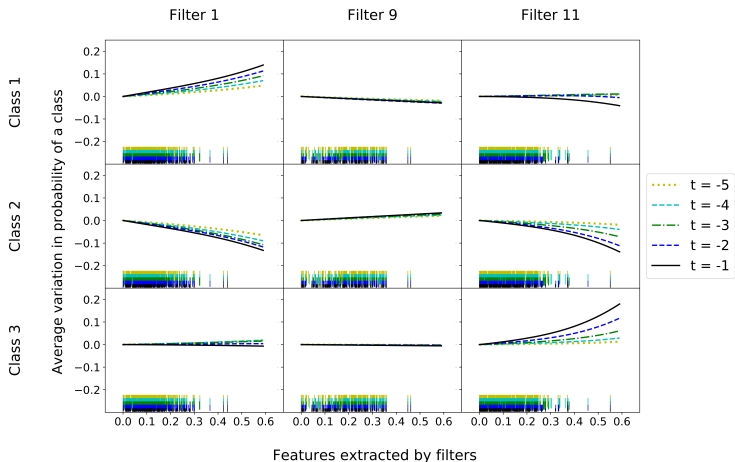
- While learning the best weights for the convolutional layer, the network aligns the filters to those tokens that help the most in the classification task;



# Similarity between filters and their most similar tokens

Filters	Tokens	$\cos(\theta)$
1	<i>final storage of docket</i>	0.46
	<i>final remittance to origin</i>	0.45
	<i>form registered in book</i>	0.42
9	<i>emitted</i>	0.47
	<i>certificate</i>	0.43
	<i>granted injunctions</i>	0.42
11	<i>temporarily stored docket</i>	0.55
	<i>docket remain in clerk</i>	0.5
	<i>return after granted period</i>	0.45




# Interpretability



# Conclusion

- Next steps: use Transformers to create better embeddings;
- Hope you liked our work. Feel free to talk to me if you want to :)
- Stay safe and with the loved ones!
- Contacts:
  - Felipe: felipemaiapolo@gmail.com
  - Itamar: itamar@tikal.tech
  - Emerson: emerson@tikal.tech

# Bibliography I

-  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** (2011), no. Aug, 2493–2537.
-  Sepp Hochreiter and Jürgen Schmidhuber, *Long short-term memory*, Neural computation **9** (1997), no. 8, 1735–1780.
-  Yoon Kim, *Convolutional neural networks for sentence classification*, arXiv preprint arXiv:1408.5882 (2014).

## Bibliography II



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