Classification de discours au Congrès des USA Machine Learning for Natural Language Processing 2020

Victoire de Richecour
MS-DS ENSAE
vdagneauderichecour@ensae.fr

Thomas Doucet
MS-DS ENSAE
thomas.doucet@ensae.fr

Abstract

We investigate whether one can determine from the transcripts of U.S. Congressional floor debates whether the part of speeches where pronounced by Democrat or Republican congressmen, as an individual's words often reveal their political ideology. Taking inspiration from recent work in text classification, we apply a long short term memory network (LSTM) framework to the task of identifying the political position evinced by a sentence. [1]

1 Problem framing

Our dataset is composed of speeches which are parts of bills debates in the Congress of the United States. Each speech can be a composed of several sentences and each one of them is tagged with a label : Democrat or Republican.

Our train dataset is composed of about 5600 samples of speeches and our test dataset possess more than 1700 of them. We will try to classify each speech according to its political orientation, with two models and two kinds of embeddings. First, we will use a Random Forest thanks to a TF-IDF embedding. Second, we will try and use a Deep Learning model : a LSTM.

The specific problems posed by this dataset are the following. Someone examining different sentences pronounced by a congressman cannot be entirely sure he is from one orientation or another (Fig 1 and Fig 2). Also, some sentences are often used in Congress, but have no particular meaning for a classification algorithm, such as : ”Mr. Chairman , will the gentleman yield ?” or ”Mr. Chairman , I demand a recorded vote.”.

2 Experiment Protocol

We first applied a naive Random Forest model with an IDF TF embedding to our train dataset. We then fine-tuned our Random Forest using a Grid Search. This improved our results. However the 2D representation of the embedding shows us that the samples are not clearly separated from

[1]https://github.com/LaDouce9/NLP-ENSAE-CongressClassification

each other in space. Even TSNE, a package that applies nonlinear transformations to the data, fails to find a separation space (Fig 3, 4 and 5).

To go further, we considered a Word2Vec embedding. We trained it on our own data (both train and test data, to have access to as much text as possible) and we also tried the pre-trained Word2Vec embedding of Google News. We then created a neural network of the LSTM type. The specificity of the LSTM is to keep in memory the most important tokens of a block of text, thus allowing the beginning of the text to keep a certain weight in the final decision. We trained our LSTM on the weights of the pre-trained Word2Vec, and on those of our own Word2Vec.

We finally evaluated the results of all these models by confusion matrices, ROC curves, and metrics such as accuracy or F1-score.

3 Results

Our Random Forest out performs the neural network by far. We achieve an accuracy of 0.73 with the random forest (Fig 6) while our best result for the network is 0.67 (Fig 7). Before running the network, we checked which embedding performed best, by averaging the weights of a W2V token and sliding them into a Random Forest. Thus, it should be noted that the Word2Vec pre-trained vectors do not allow us to improve our results since the accuracy of this model (pre-trained W2V + RF) only reaches 0.53. On the other hand, it allowed us to realize that a Word2Vec model trained on our text corpus was a better embedding. We therefore chose this embedding to launch our neural network.

4 Discussion

Several elements can explain these different results. First, our Word2Vec embedding does not have enough text to train properly. In order for it to better understand the relationships between words, it should be provided with more samples. The pre-trained Word2Vec does not understand the tokens of our vocabulary in their context. Indeed, the Google News corpus is trained on Wikipedia and other open-source sites, but the particularity of congressional speech escapes this Word2Vec model. Unfortunately, we could not fine tune a pre-trained Word2Vec model due to our limited memory resources. Moreover, our LSTM network is probably not tuned well enough to match Random Forest. Yet we have gone from a model that did not learn at all to one that learns significantly more than before.

In conclusion, neural networks require much more time and investment to achieve a result that is sometimes only slightly better than machine learning algorithms such as Random Forests. In terms of time cost, a random forest in way faster than a LSTM since our LSTM took some fifteen minutes to run. In terms of space complexity, the use of our own trained Word2Vec allows a gain in memory compared to Google News Word2Vec.

If we were to continue to improve our model, we've been thinking about a few things. First, use the pre-trained Word2Vec vectors from Google News, and re-train them ourselves to gain a finer understanding of the text in its context. Second, after continuing the fine-tuning of our LSTM, add a layer of attention to give more weight to the most important words in each sentence.

Appendix

WordClouds

Figure 1: WordCloud for the whole train dataset

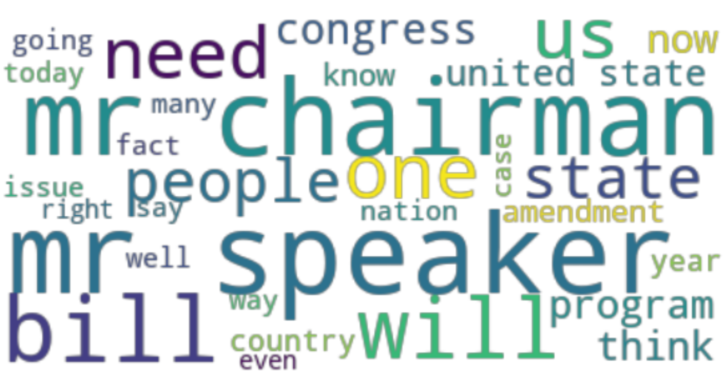
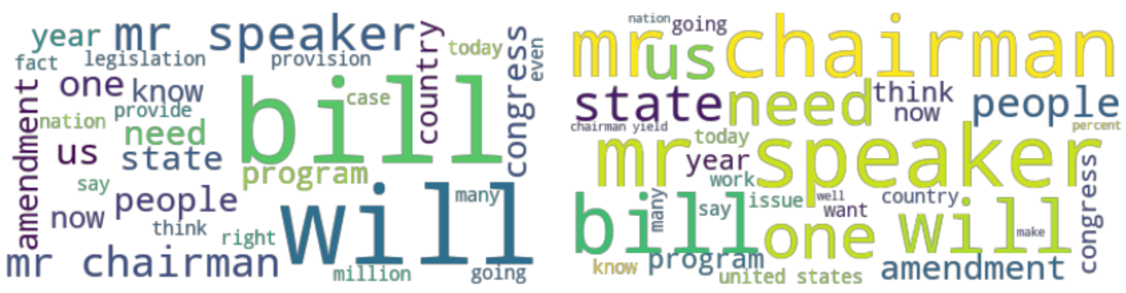


Figure 2: Republican and Democrat wordclouds



2D representations

Figure 3: 2D representation of TF-IDF embeddings

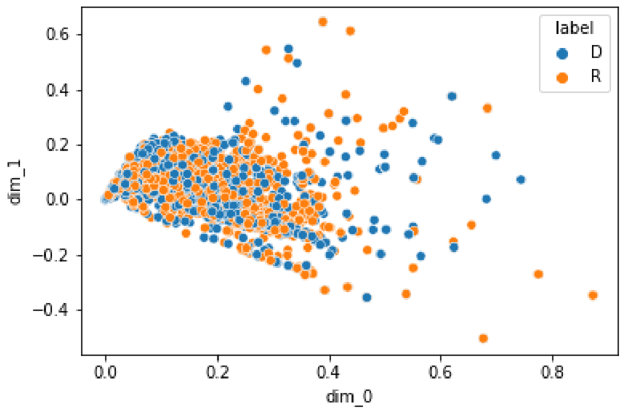


Figure 4: 2D representation of TSNE embeddings

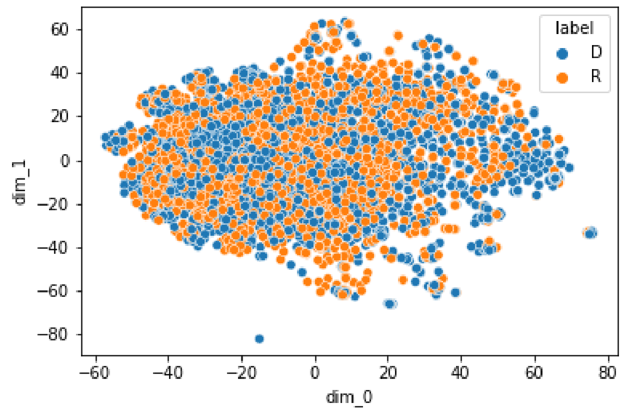
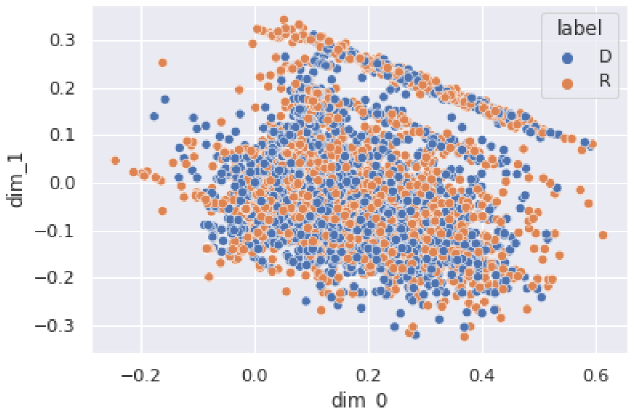
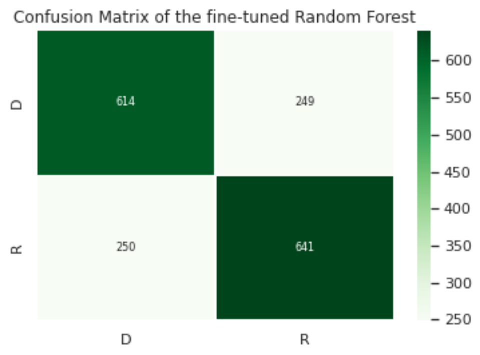


Figure 5: 2D representation of W2V embeddings



Random Forest results



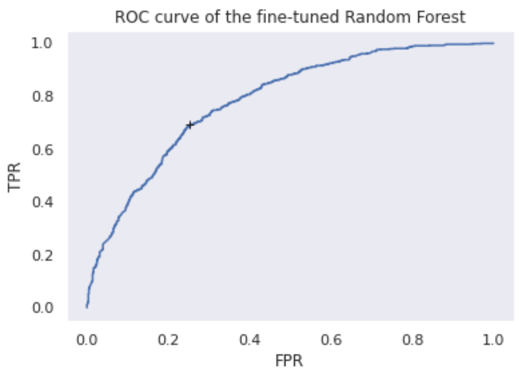
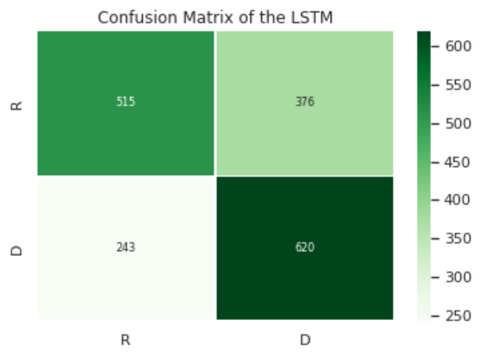


Figure 6: Random Forest results

LSTM results



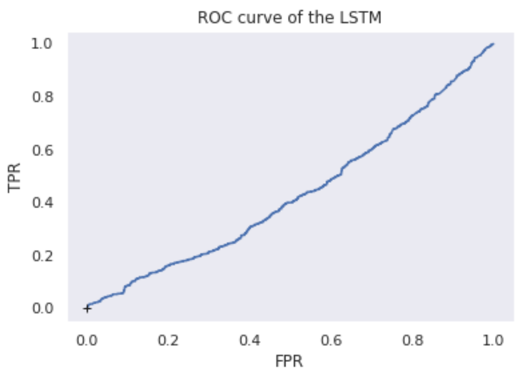


Figure 7: LSTM results