Natural Language Processing

REPUBLICAN VS. DEMOCRAT
TWEET CLASSIFICATION

Clément Detry, Karim Eschweiler, Selim Gilon, Aaron Schapira, Adèle Imparato

May 28th, 2021

Abstract

This report describes algorithms that enable to classify someone's political opinion, more specifically, whether the person is Democrat or Republican. This information is valuable in numerous contexts described in this paper. In order to classify people, the first step is to get a labeled database such as scraped tweets from some American politicians. Then, to analyse those data, this report will focus on several techniques such as preprocessing methods and supervised classifiers (Naive Bayes, Logistic Regression, Decision Tree, Random Forest, k-Nearest Neighbors and LSTM Neural Network).

Eventually, this will lead to the conclusion that, indeed, a machine learning model is able to classify the political party of a politician following the techniques implemented in this paper. Furthermore, the best accuracy reached for this task is $\pm$ 95% using the random forest classifier, due to its capacity of computation. For both labels, one can observe a certain vocabulary specific to the individuals of this class.

1 Introduction

Having an efficient text classifier that defines someone's political opinion based on public tweet text could be extremely useful. More specifically, this could be a classifier that decides whether someone is more Democrat than Republican. For example, nowadays, a significant number of firms and enterprises are interested in labeling internet users in order to recommend them targeted information. This information could be sent to users as an advertisement, but also in the form of first research results when they google something. This way, recommendation systems on the internet could influence users regarding their political opinion, by suggesting them content that is similar to their opinion or, inversely, varied content to qualify their opinion. This is actually what happened during the US presidential elections in 2016: recommendation systems had influenced people's vote, which caused a scandal. A better application of this mechanism would be to predict the political opinion of a certain US state based on the tweets published during the last months preceding the elections.

For humans, it could be tough to identify a person's political opinion based on a post written in a few sentences. Sometimes, it may be impossible, because of the human brain which does is not 'powerful' enough for the amount of statements made. But a computer could manage to try identifying a person with the best accuracy as possible using Natural Language Processing (NLP). This is what is going to be experimented in this report which is based on two major research questions:

1. Can a machine learning model predict the political party of a politician based on his/her social media's posts? What is the best classifier and what is its accuracy ?

2. Furthermore, can this model be used for a social media user which is not a politician ?

3. Can some specific vocabularies be created for the two different classes ?

2 Related work

As one might imagine this subject has already been treated by several researchers such as Jon Reynolds is his article "Predicting Political Affiliation with Natural Language Processing" [6] or even Kamal Chouhbi in his article "Are you Democrat or Republican? Let your tweets define you ..." [3] and many others.

In his paper, Reynolds discusses the different techniques he used for his task: to predict whether the author of a forum's post is more aligned to the Democratic of Conservative parties. He mainly used the tool Sci-Kit Learn as well as other well known machine learning Python's libraries and some techniques of classification modeling such as

NLTK, Gridsearch, TF-IDF, CountVectorizer, Logistic Regression, Random Forest, AdaBoost, and Multinomial Naive Bayes.

On his part, Chouhbi aims to answer the question "Are you a Democrat or a Republican?" by analyzing tweets. To do so, he used several well known techniques such as: preprocessing, sentiment analysis, machine learning and a neural network (RNN).

The techniques used by these two researchers are the most widespread ones. This is why some of those have been used in this report, such as TF-IDF, Logistic Regression, Naive Bayes, Random forest, the LSTM Neural Network and some prepocessing methods. Moreover, this report describes the decision tree model and eventually the technique of the k nearest neighbour.

3 Methods

3.1 Data

3.1.1 Data Scraping

In order to predict whether the author of a small text input is Democrat or Republican, one needs a data base of inputs, in this case, tweets. To this end, 23208 tweets have been scraped from the Twitter API.

Scraping tweets directly from the online platform requires a Twitter Developer account, which can be obtained after sending a request to Twitter. Afterwards, the process is quite simple, it only requires a few lines of code in Python to extract the data. Then it demands some extra time in order to re-structure the extracted tweets in a pandas dataframe.

3.1.2 Data Description

The data then consists of: the tweet, the date of the tweet, the author's pseudo, the author's name, the id of the tweet, the label (Democrat or Republican) and eventually the tweet's length. The only way to get the labels for this specific task is to scrape tweets from politicians for whom their political party is known.

It is important to notice the balance of the data. As it can be observed in Figure 1 below, there are slightly more tweets labeled as Republican than Democrat. This should not lead to any issues since the difference is minor (Democrat: 43%, Republican: 57%). Otherwise it will require some data balancing before training classifiers on it.

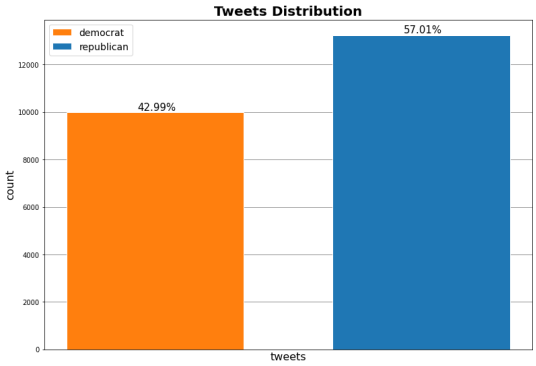

Figure 1: Data tweets distribution

3.2 Data Preprocessing

In order to have a better data to work with, there are different ways to preprocess it. Here are some of the approaches used in order to have relevant features for the multiple analysis.

- Lower-casing is useful to group words that are written with different letter case but still have the same spelling.
- Remove every possible symbols in the tweet texts.
- Get rid of English stop-words. They are too often mentioned but do not add a lot of meaning in the tweet.
- Remove links provided in a tweet to allow a convenient parsing.
- Lemmatizing to nouns, verbs, adjectives, and adverbs. This reduces the original words to common roots and therefore, reduces the size of the vocabulary.

After preprocessing the data, the vocabulary size is about 24000 whereas without the preprocessing step, its size is about 72000 [13]. Having a smaller vocabulary size is good because it allows not to take useless tokens into account. For instance, in the initial vocabulary, the word "issue" was written in about 30 different ways ("issues", "isssues", "isue",...), all part of the vocabulary. After the preprocessing step, it got reduced to one single token ("issue"). This way, spelling mistakes and similar words are avoided and simply reduce to their lemma.

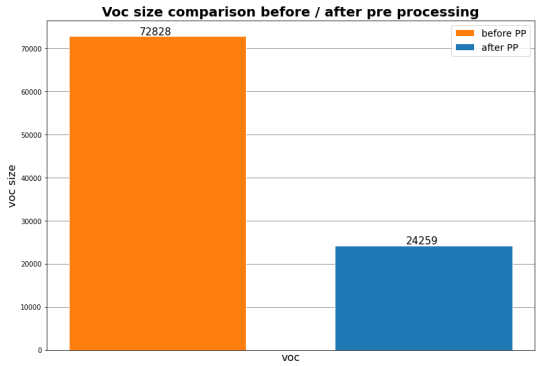

Figure 2: Voc size before and after preprocessing tweets

3.3 Filtering

Another step that was done is filtering the data. This step enables, again, to reduce the size of the vocabulary by filtering out tokens depending on their frequency. The goal of this step is to remove the least frequent words. The motivation behind is that, words which occur just a few times in thousands of tweets are probably not that useful to classify the tweet. An experiment will be performed in order to observe the effect of the threshold on the classifiers (see Figure 6).

3.4 Feature Creator

To make the machine learning models operational on the data set, they need to understand the representation of words inside tweet texts as real number vectors. Two different techniques are used to represent those vectors: Bag of Words and TF-IDF (Term Frequency – Inverse Document Frequency). Both of their efficiencies will be compared.

3.4.1 Bag of Words

This feature creator technique consists of two major methods:

1. First one creates the feature set by instantiating a vocabulary with occurrences of words in the set of tweet text. In parallel, it creates another vocabulary that stores the id number of each feature word of the data set.

2. Second one design the feature matrix that will be used to train the classifier. This feature space has a size of the length of the data (total number of tweets) x the length of the previously computed id vocabulary. Each tweet consists of a horizontal vector in the matrix. The value at a certain word id position is 1 if this particular word feature is in the corresponding tweet, otherwise this value is 0. This matrix also contains the label of the matching tweet at the end of each row. If this value is 1, it means the tweet is Republican, otherwise it has been published by a Democrat.

   This following example should clarify it. Each feature word is written as $w_{i,j}$, where i stands for the tweet number and j for the word id index. In addition, l_i corresponds to the label of the i^{th} tweet.

$$\mathbf{X} = \left[\begin{array}{ccc|c} w_{1,1} & \dots & w_{1,m} & l_1 \\ \vdots & \ddots & \vdots & \vdots \\ w_{n,1} & \dots & w_{n,m} & l_n \end{array}\right]$$

3.4.2 TF-IDF

As another way to create the vectors representing the tweets, the tf-idf algorithm was implemented (term frequency - inverse document frequency model). This model helps to know what the importance of a word in a tweet is. It is used as a weighting factor in searches of information. The tf-idf value increases proportionally to the frequency of the word.
The implementation is as follows. First, the value of the term frequency is computed with the following formula: (number of times the word appears in a tweet over the total number of words in the tweet)

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

Then, the inverse data frequency is calculated. The formula can be described as the weight of a word, since the number of tweets is divided by the number of tweets containing the word. The computation is as follows:

$$idf(w) = \log \frac{N}{df_i}$$

The final step would be to compute the combination of the two scores to get the final TF-IDF score for a word in a tweet.

$$w_{i,j} = tf_{i,j} * idf(w)$$

After all those calculations, the vector for a word is done and can be fed into the different Machine Learning models.

3.5 Machine Learning Classifiers

In order to have a well trained classifier, one can perform an analysis that allows to select features that could be relevant to distinguish both classes: Democrat and Republican.

As a first step, one can use a tool to find out the most occurring words in both classes in the training set. To do so, it is important to choose a threshold, which will determine what are the relevant frequencies of words that would create consistent dictionaries. Those dictionaries would then be fed in the different statistical machine learning models.

3.5.1 Naïve-Bayes classifier

This classifier is based on the Bayes Theorem for calculating probabilities and conditional probabilities to predict the class of unknown data sets (**Generative** model).

One of the advantages of this model, is that from the prior probability, the probability of a related event can be computed. This has a good asset in terms of classification and predictive modeling for such our problem.

The fundamental Naive Bayes assumption is that each feature makes an independent and equal contribution to the final outcome :

$$P(y|x_1, ..., x_2) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)}{P(x_1)P(x_2)...P(x_n)}$$

This means that the algorithm will look at the individual words in the different tweets and not the full sentence itself. Even though the different neighbors of a word influence its meaning, the order of the words in the sentence does not matter.

Regarding the classification problem, the class variable y has the outcome "Democrat" or "Republican". Furthermore, to be able to predict the label of a tweet, the class with the highest posterior probability will be the outcome. This process would then be repeated for every tweet of the training model. [2]

3.5.2 Logistic Regression

Another Machine Learning algorithm used in this report is the Logistic Regression (LR). Similarly, it can be used to classify an observation into one of the two classes.

In contrast with the NB classifier, Logistic Regression is a linear classification method that learns the probability of an instance that belongs to one of the two classes (Democrat & Republican). Its goal is to find the perfect boundary that separates the data into classes as good as possible.

Once the model is trained enough, the algorithm is able to identify the proper classes with the performance of the following table. The lowest the cost, the better is the algorithm.

3.5.3 Decision Tree

A decision tree is represented in a flowchart tree structure way with nodes and edges (cf. Figure 3). There are two types of nodes contained in this type of framework: internals and leaves ones. An internal node stands for a particular feature in the data set and a leaf aims to define a specific outcome. Between them are the edges that can be considered as branches that represent a decision rule.[5]

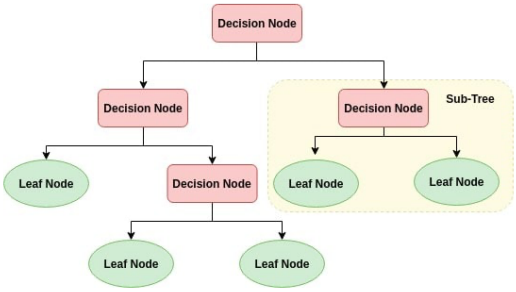

Figure 3: Decision Tree Visualization[5]

Decision tree is a very intuitive and easy classifier to interpret classification tool. Building it is not really much harder than reading one. The basic idea behind its implementation is the following (cf. Figure 4):

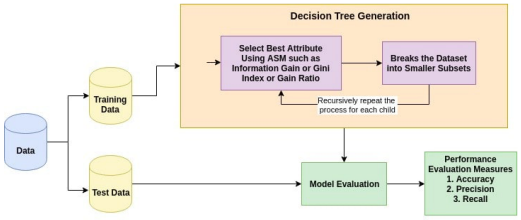

Figure 4: Decision Tree Training[5]

1. It selects the optimal predictor on which to split as well as the the best threshold value for splitting it.

2. Then, it transforms this node into a new decision attribute and breaks the data set into smaller subsets of information.

3. These two first steps are recursively repeated until a predefined stop condition is met.

The advantages and disadvantages of using a decision tree as a classification technique are be the following[1]:

+ Cheap computational cost to construct

+ Easily interpreted model (as long as they remain small)

+ Removes meaningless features

– Tends to overfit training set

– Very sensitive to variation in data set

– Restricted decision boundary

3.5.4 Random Forest

The random forest classification technique can be viewed as an 'upgrade' of the decision tree one,

previously described. It consists of a large number of individual decision trees that perform as a cluster[8]. Each single tree in the constructed random forest makes its own class prediction and the one with the greater number of votes becomes the chosen model's prediction (cf. Figure 5).

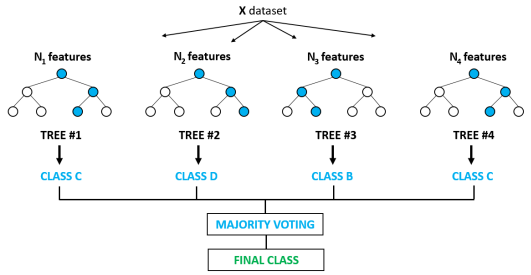

Figure 5: Random Forest Visualization[7]

What are then the pros and cons of using such a classification technique instead of the basic decision tree one[4]:

+ Stable model against new data point introduced
+ Very slightly impacted by noise
+ Reduces overfitting
– Expensive computational cost
– Long training period

3.5.5 k-NN

The Nearest Neighbors technique is another Machine learning classification algorithm. To be able to estimate a class value y for a given test instance x (in this case a tweet), it will find a set NN of the k closest instances to x in training data D. In this context, a scoring classification is done: the classifier will output a score for each class among the instances of the NN. The score are then normalized and the class probability is estimated.

Different values can be used for the k. This value controls the complexity and locality of the decision border of the k-NN classifier. The smaller the value is, the more complex is the decision border of the classifier since it becomes more local.

3.6 LSTM Neural Network

Long Short Term Memory networks (LSTM) are a specific kind of Recurrent Neural Network (RNN) that can overcome the issue of learning long-term dependencies. The particularity of these networks is that they consist of four NN layers (instead of usually one) forming a chain of repeating modules. These four layers interact together in a very specific way.

4 Results & Discussion

This section contains the results that have been conducted on the tweet data.

4.1 Filtering

The filter shows that the words with a small frequency are not useful for the classifier. This can be explained by the fact that if a word has a low frequency in the training set, it will most likely not occur in the test set.

When having a deeper look on the graph, one can see that the most reasonable configuration would be to have a threshold that has a value between 10 and 100. In this range of values, it can be seen that the accuracy is still pretty good even though up to 80 percent of the vocabulary is removed (the different percentages of the vocabulary left after the filtering can be seen in Figure 6). However, when looking at the accuracy for a bigger threshold (represented by 500 in Figure 7), there is a significantly decrease of the accuracy.

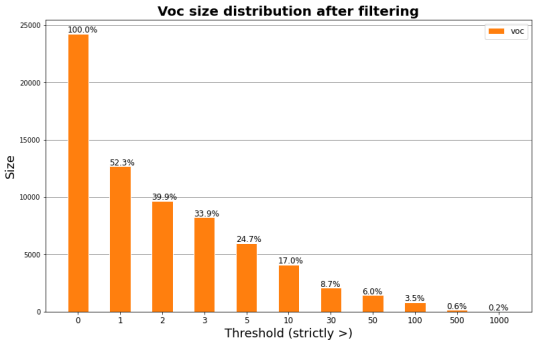

Figure 6: Different thresholds for filtering.

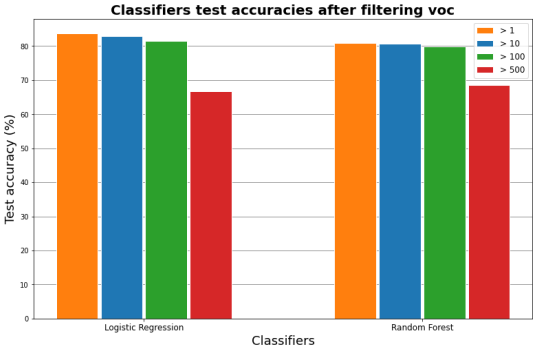

Figure 7: Classifiers test accuracy after filtering

As Figure 8 and Figure 9 show, filtering words that have a small number of occurrences clearly improve the efficiency of the model. That's the major advantage that can be took of filtering the vocabulary on the training data.

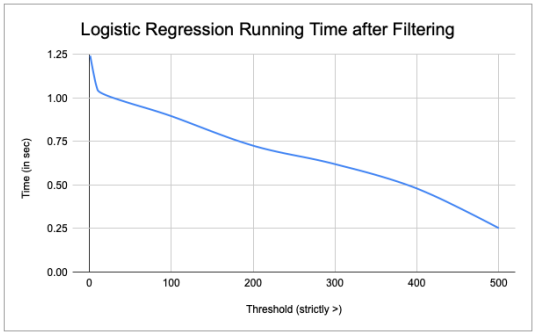

Figure 8: Running time after filtering - Logistic Regression

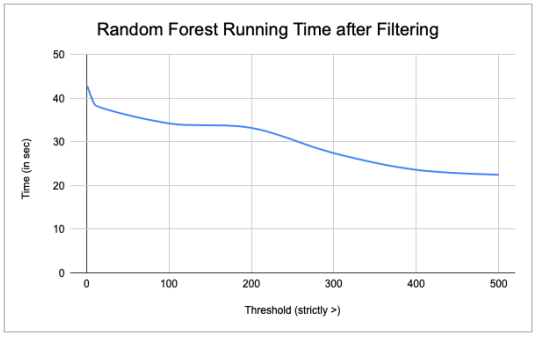

Figure 9: Running time after filtering - Random Forest

4.2 Accuracy

Firstly, the individual scores of each classification technique previously described with different configurations. Then, more findings insights about the most relevant ones. Finally, some other interesting vocabulary pattern discovered for each class by manipulating the data.

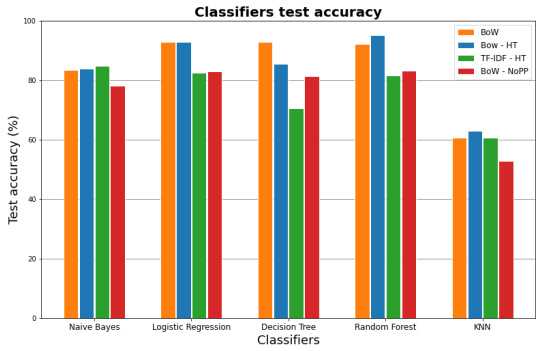

Figure 10: Machine Learning model's accuracy

From these results we can observe that the Logistic Regression and the Random Forest model perform best. Indeed, the Logistic Regression classifier reaches an accuracy above 80% in all four categories (BoW, Bow-HT, TF IDF-HT, BoW-NoPP), which is rather high. Similarly, the Random Forest model reaches an accuracy of more than 80% in all four sections as well. Compared to the other techniques used, it is clear that these two models are the most accurate.

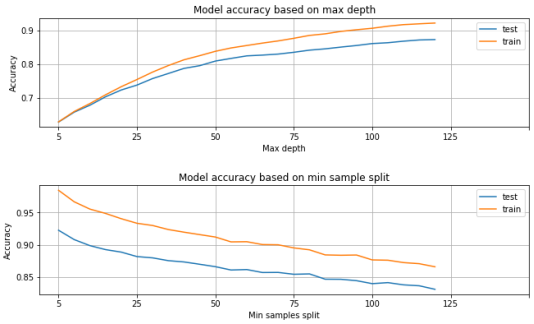

Figure 11: Decision Tree train-test accuracy

On the first graph above, one can observe that as the depth of the tree increases, the training accuracy increases as well. In fact, when adding more and more leaves, the model gets closer to an overfitting scenario. Indeed, the training accuracy increases as the model catches all the words. In this scenario, one would expect the testing data to be less accurate but in this case the accuracy increases as well, this might be due to the fact that the training data is very similar to the testing data. Another hypothesis is that the highest nodes of the tree are rapidly splitting the data into two parts: the Democrats' vocabulary and the Republicans' vocabulary.

On the second graph above, one can observe that as the value of the min sample split increases, the amount of branching diminishes and both accuracies decrease with the same behavior. Therefore, the best value for the min sample split is 5 as it gives the best accuracy.

Eventually, one can conclude that the decision tree model is adapted to this specific problem as it reaches an accuracy close to 0,9. However, it focuses only on politic-related text inputs. Therefore, this model would not apply to the similar problem but analyzing random text inputs.

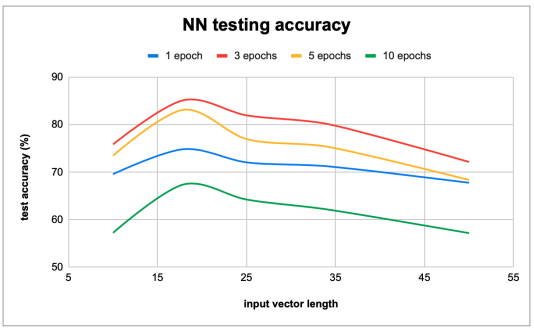

Figure 12: NN test accuracy

Figure 11 indicates the testing accuracy of the

NN model on the set of data while tuning some parameters. First, the size of the input vectors can be modified. It means that if the tweet sentence contains less tokens than the selected size, its corresponding vector will be filled in with 0 values. Generally, it is better to have a pre-selected size of vectors that best suits the data. In the preprocessed data, the tweet with the highest number of words in it is 34. The mean of entire set of tweets is 18 and this is the best value we can set for it (cf. Figure 12).

The total number of training in the NN is also a parameter to tune. After testing different choice, the best number of passage through the neural network is 3. Under this threshold, it tends to underfit the data, otherwise it overfits it.

4.3 Frequent words

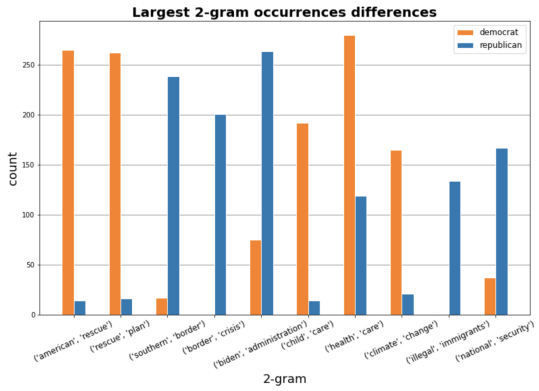

Figure 13: Data largest 2-gram freq differences

On figure 13 we can see the sequences of two words having the largest difference of occurrences. The ten most ones have been selected using the following formula:

$$\Delta(w_i, w_j) = |democrat[w_i, w_j] - republican[w_i, w_j]|$$

,where (w_i, w_j) are two following words in tweet texts (= 2-gram).

The major observations that emerge from this graphic is that Democrats have a certain preference for discussing these topics: American rescue/rescue plan, child care, health care, climate change, etc. while Republicans prefer tweeting about: the Southern border, the border crisis, Biden administration, illegal immigrants, national security, etc. These topics indeed coincide with their respective ideologies.

5 Conclusion

In this work, some natural language processing techniques to process social medias' posts have been applied. The goal was to use these to classify the author's political party. The dataset used in this report consists of 23208 scraped tweets from the platform Twitter. The labelling was made by hand but this was not challenging as the tweets were scraped from popular American politicians. To achieve this goal, some popular preprocessing steps have been used before building and training different supervised classifier such as Decision Tree, Logistic Regression, Naive-Bayes, Random Forest, k-nearest neighbors, and LSTM NN.

Filtering the vocabulary with a minimum number of occurrences radically decreased the vocabulary size (3.5 % left with a filter size of 100) while maintaining a high accuracy (about 80%) on the classifiers. Eventually, this filter accelerates a lot both the training and the testing steps of the the classification. Also, the lemmatization technique divided by 3 the vocabulary size.

Regarding the features engineering methods, the bag of words is usually a better option than the TF-IDF.

The best testing accuracy that has been reached is 95% with the tuned random forest and a more or less balanced dataset (43% - 57%). The only classifier which has a best accuracy less than 80% is the K-NN.

However, a tree based ML model might not be the best option in case we would like to extend this classification problem to some lambda person which are not politicians and post a lot of unrelated tweets. and

The best accuracy with the LSTM NN is hit by using 3 epoch and 18 as features vectors' size.

We also observed some significant differences of word occurrences based on the political party.

6 Further improvement

In order to answer to one of the research questions of this paper, namely "Can this model be used for a social media user which is not a politician ?", one could argue that it is not possible because of the training set used. Indeed, the training set contains only tweets from politicians, thus politic-related texts. Based on that, testing inputs from an average person discussing any random topic wouldn't allow the classifier to predict the proper label. Also, even if it could manage to predict the political party of the author, there won't be a way to check whether the prediction is correct. Therefore, the only possible testing set is one filled of politic-related input texts from already-labeled authors.

Otherwise, to be able to predict the political party of the author of any random input text, it would require the model to have a training set full of multi topic texts, each of them labeled correctly. Assuming the training set is large enough, the model would eventually be able to predict the proper label whatever the topic of the input text. But again, this will require to label lots of text in-

puts which is almost unfeasible since the political opinion of a lambda person is not public/known.

References

[1] A. Chakure. *Decision Tree Classification. An introduction to Decision Tree Classifier*. 2019. URL: https://medium.com/swlh/decision-tree-classification-de64fc4d5aac.

[2] Nagesh Singh Chauhan. *Naïve Bayes Algorithm: Everything you need to know*. 2020. URL: https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html.

[3] K. Chouhbi. *Are you Democrat or Republican? Let your tweets define you...* 2020. URL: https://towardsdatascience.com/are-you-democrat-or-republican-let-your-tweets-define-you-4aa4cadf4bea.

[4] K. Naresh. *Advantages and Disadvantages of Random Forest Algorithm in Machine Learning*. 2019. URL: http://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html.

[5] A. Navlani. *Decision Tree Classification in Python*. 2019. URL: https://www.datacamp.com/community/tutorials/decision-tree-classification-python.

[6] J. Reynolds. *Predicting Political Affiliation with Natural Language Processing*. 2019. URL: https://medium.com/@jon.reynolds30/comparing-classification-models-using-lending-clubs-peer-toloan-data-615153db2124.

[7] N. Smith. *Data Science Portfolio. Random Forest*. 2019. URL: https://najeesmith.github.io/RF/.

[8] T. Yiu. *Understanding Random Forest. How the Algorithm Works and Why it Is So Effective*. 2019. URL: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.