EURECOM
Sophia Antipolis

MALIS Project : FAKE NEWS DETECTION

Yassine ELKHEIR
Data Science and Engineering
yassine.elkheir@eurecom.fr

Prince AMANKWAH
Internet of Things
prince.amankwah@eurecom.fr

Abstract

Our modern society is struggling with an unprecedented amount of online misinformation, which does harm to democracy, economics, and national security. Since 2016, Fake news has been well tuned and aligned with United states elections, and that came back to the media recently with the approach of elections, so we tried though that project, based on the highly motivation nowadays “US Election” to develop a reliable model that classifies a given news article as either fake or true. We tried to simulate the different programs made by large big companies, like google, on sources and web sites as well. Our simple models give us accuracy up to 76%.

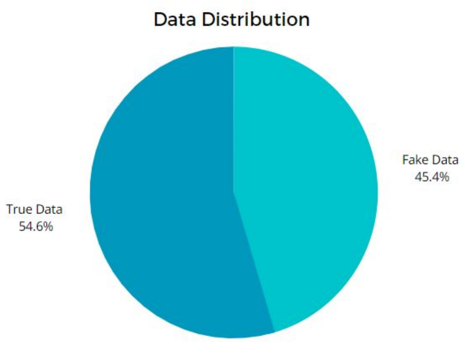
Key Words *: Bag of words, TF-IDF, Doc Vec, Naive Bayes, Logistic Regression, SVM, Random forest, MLP.*

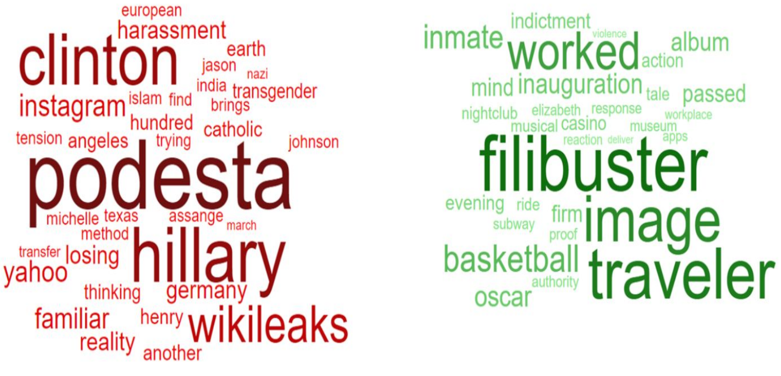
Introduction

In a common view, that problem “ Fake news ” is increasing especially with that huge growth of social media. This growth of media which costs low, easy to access, and rapid destination of information lead people to seek out and consume news from social media . As a result, detecting and automating fake news detection is essential to maintain a robust network and a safe social media in general. With this motivation, through building our knowledge of python and machine learning, we worked on our final project. The input of our algorithms is a text, which goes initially through a fundamental step, in our prospect to define a perfect final model, which it is “processing”, in order to vectorize it, and making it digital (in matrix format), for this step, we’ve used until now three methods, Bag of Words, TF_IDF, and Doc_to_Vec, after fitting our models, we intend to output a 1/0 value to predict if the text tested is TRUE or FAKE.

SYSTEM ARCHITECTURE & ALGORITHMS

As mentioned in the problem statement, the project deals with identifying fake news from the given dataset based on the 2016 US election, an extract from kaggle. Which is a dataset containing different texts (taken from articles) with a label determining if the article is true or false. Size of the training set is about 40000 **tokens**, and the size of the testing set is about **5193 tokens**. The implementation involves tasks such as data preprocessing, feature extraction, training models etc.





(a) Fake news tokens

(b) True news tokens

Methods

1. Clean data set :

This class aims to remove empty texts from our training set, put all words in lowercase, remove prepositions and “Doesn’t affect” words using stemmer(“english”), remove words of size less than 3, and finally remove punctuations.

2. Feature Extraction :

For the fake news detection, the actual news data (body of the news article) is being considered as features. But the data is in the form of text. It is known that for machine learning analysis, text data does not work well. So the text data has to be converted into a numerical representation. This process is called vectorization. Every record (i.e news article in this case), after cleaning data inside it, should be converted into a vector. There are several techniques which can convert text to vectors. Below is the list of such techniques that we pursued :

Bag of Words :

A bag-of-words model is a way of extracting features from text for use in modeling, such as with machine learning algorithms. The approach is very simple and flexible, and can be used in a myriad of ways of extracting features from documents. A bag-of-words is a presentation of text that describes the occurrence of words within a document. It involves two things;

1. A vocabulary of known words.
2. A measure of the presence of known words.

No one can deny, that is the simplest way to vectorize a record, but the main challenge in this technique is that all the words (among the ones you chose) are weighted uniformly which is not true in all scenarios as the importance of the word differs with respect to context.

TF-IDF : Term Frequency – Inverse Document Frequency

In this model, the words are assigned a weight based on the **frequency of appearance**. The model has 2 parameters. The term frequency component adjusts the weight proportionally with the number of times the word appears in the document with respect to the total number of words in that document.

Inverse document frequency component identifies unique words in the set of documents and increases weight accordingly. If a particular word is appearing in most of the documents, then its weight is reduced as it will not help anyway in distinguishing the documents. Though this model weighs the words based on the frequency and unique factors, it is not able to capture the meaning of the word.

Doc2Vec :

For the previous models, the context of the word is not taken into consideration. The same word appearing at two different locations in the same text convey different meanings. Doc2Vec takes in consideration the **context** of the word. This is a neural network driven model and is an improvement over the previous models. Doc2Vec models take local context into consideration, the surrounding of the word. This model, it’s stronger than Word2Vec, as an input of the neural network takes an id which represents the record in which that words appratient, not like Word2Vec which fails to recognize the global context.

3. Classification Model

Naive Bayes :

The Naive Bayes classifier assumes that each element of the input vector is statistically independent. During training process, we first use training samples that are labelled ck (k = 0, 1) so that the model can fit the following probability distribution:

$$P(x_i \mid c_k) = \frac{1}{\sqrt{2\pi\sigma^2_{c_k}}} \exp\left(-\frac{(x_i - \mu_{c_k})^2}{2\sigma^2_{c_k}}\right)$$

where xi is the element of each input vector (each model of extraction has a number). During prediction process, given an input vector x = (x0, x1, ..., xn) T , the model predict the class it belongs to use the following equation:

$$\hat{c_k} = \operatorname*{argmax}_{c_k} P(c_k) \prod_{i=0}^{n} P(x_i \mid c_k)$$

Logistic Regression :

Logistic regression models the relationship between features and the response variable, which in this case is the truth of our text (record), through the logistic function, which takes the form:

$$h_\theta(X) = \frac{1}{1 + e^{-\theta^T X}}$$

X is the features corresponding to a given record, and θ is the parameter that our model learns during training. This outputs a vector with values [FAKE, TRUE], which we then get our prediction from by taking the softmax of this vector. To optimize this parameter, we perform l2 regularization by minimizing the following cost function:

$$\min_{\theta, c} \frac{1}{2}\theta^T\theta + C\sum_{i=1}^{n} \log(\exp(-y_i(X_i^T\theta + c)) + 1)$$

SVM Support Vector Machine :

The SVM algorithm uses a hinge loss that seeks to maximize the margin between the two classes of data. The SVM algorithm uses a second-order Gauss kernel that operates on the full token features (done by previous models) space. The expression for this kernel is given by the following expression:

$$G(x;\sigma) = \frac{1}{\sqrt[2]{2\pi\sigma}} exp(\frac{x^2}{2\sigma^2})$$

Note that this expression is provided for the 1-D case. In retrospect, the selection of this high-order kernel seems rather naive, since it may have caused the SVM model to over fit the training set.

K-nearest neighbors algorithm :

kNN is considered a non-parametric method given that it makes few assumptions about the form of the data distribution. This approach is *memory-based* as it requires no model to be fit. Nearest-neighbor methods use the observations from the training set closest in input. It is based on the assumption that if a sample's features are similar to the ones of points of one particular class then it belongs to that class. These points are known as nearest neighbors.

Multi Layer Perceptron :

MLP Classifier trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters.

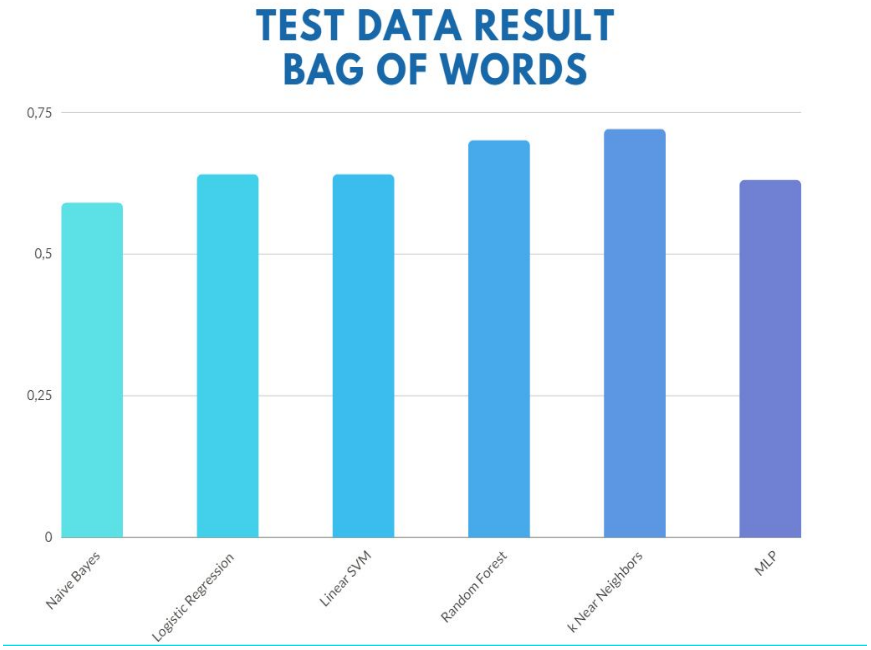
Random Forest Classifier :

A decision tree is a “tree” where different conditions branch off from their parents and each node represents a class for classification. The random forest classifier is an ensemble method that operates a multitude of decision trees and thus improves the accuracy. We adjust parameters such as max depth, min samples split, n estimators, and random state to achieve the best performance; where Max depth is the maximum depth of a decision tree; Min samples split is the minimum amount of samples to split an internal node, and N estimators is the number of decision trees in the random forest.

Evaluations

Bag of Words Model :

Using the Bag of words model to extract features, we observed an accuracy of 74% percent on our testing set, using k_near neighbors algorithm model.

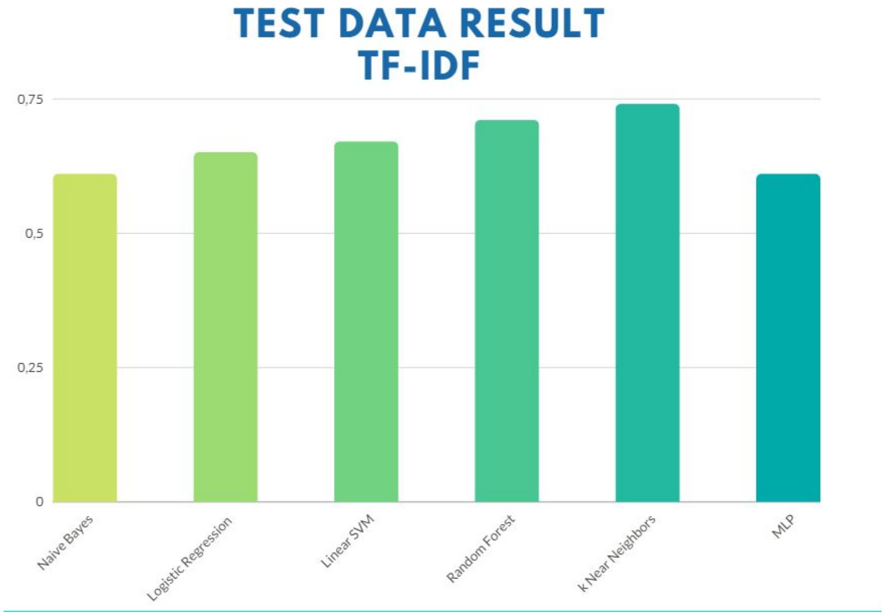


Result :

Naive Bayes	:59%
Logistic Regression	:64%
Linear SVM	:64%
Random Forest	:70%
kNN	:72%
MLP	:63%

TF-IDF :

The increase in accuracy for the models made previously was foreseen, since the TF-IDF method does not equalize all the inputs (as in the bag of words method), but weights them according to their appearance in the inputs. As last time, the highest accuracy was reached by k Near Neighbors for an accuracy of 74%

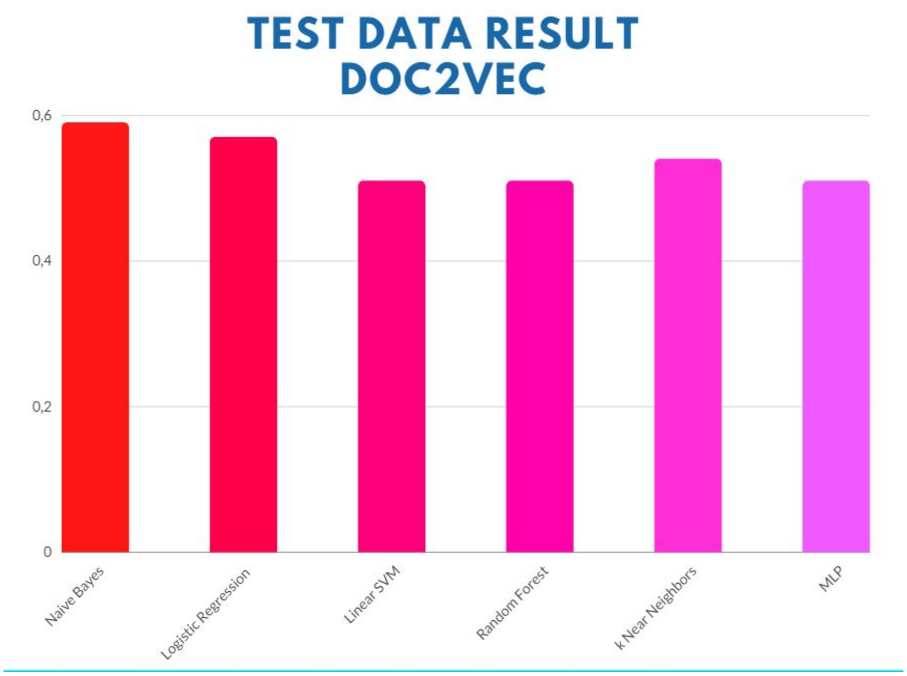


Result :

Naive Bayes	:61%
Logistic Regression	:65%
Linear SVM	:64%
Random Forest	:71%
kNN	:74%
MLP	:61%

DOC2VEC :

We learned that the most frequent tokens are not the most casual ones, so the context of the word that Doc2Vec takes into consideration in fitting doesn’t affect a lot, unfortunately that drops accuracy.



Result :

Naive Bayes	:59%
Logistic Regression	:57%
Linear SVM	:51%
Random Forest	:51%
kNN	:54%
MLP	:51%

5 . Contributions :

1 : Preparation of data : Prince AMANKWAH
2 : Search for other datas to reinforce our models and adapt it to the initial dataset
3 : Extraction of Features Models : Yassine ELKHEIR
4 : Algorithms of models : Yassine ELKHEIR
5 : Test our Models : Yassine ELKHEIR (Bag of Words & TF-IDF) & Prince AMANKWAH (Doc2Vec)
6 : Final Report : Yassine ELKHEIR
7 : Update Report : Prince AMANKWAH

Resources :

- Eurecom Malis Courses (Maria Zuluaga)
- Doc2Vec : https://cs.stanford.edu/~quocle/paragraph_vector.pdf
- Fake_news_detection : http://cs229.stanford.edu/proj2017/final-reports/5244348.pdf
- MIT Thesis : https://dspace.mit.edu/handle/1721.1/119727
- Research Article :Fake News Detection Based on Machine Learning by using TF IDF :https://www.ijesc.org/upload/637dc1f20df31f7aa5680e96eed92665.Fake%20News%20Detection%20Based%20on%20Machine%20Learning%20by%20using%20TFIDF.pdf