KTH Royal Institute of Technology

Language engineering

Final Project Report

Fake news detection

Giovanni Castellano Yassine ELKHEIR

October 3, 2022

Abstract

Our modern society is struggling with an unprecedented amount of online misinformation. In this project we develop a model that classifies a given news as either fake or true. We use different models (naive bayes, logistic regression and LSTM) together with different embedding algorithms (Bag-of-words, Term Frequency — Inverse Document Frequency and Word2Vec). The dataset used during the project contains 45000 news which are splitted in order to get a training and a test set. To measure the goodness of the models we use F1-Score and accuracy. The project is implemented in python and the most important used libraries are NLTK, Scikit-learn, and Tensorflow. The results show that all tested models give very high predictive accuracy, with LSTM that outperforms others. Results also show that different embedding methods can affect the performance of the models.

Keywords— Nature Language Processing, Fake news detection, LSTM, Bag of Words, TF-IDF, Word2Vec, Logistic Regression, Naïve Bayes, Biased Dataset

I Introduction

Fake news is one of the biggest issues of journalism. Every day thousands of fake news spread through the internet causing panic among people. During the coronavirus pandemic for instance, many people were skeptical of the idea of getting vaccinated due to the misinformation generated by fake news. Influential people such as politicians also exploit fake news to gain consensus among people and this has serious consequences for society. Compounding the problem is the fact that people tend to trust any news that confirms their beliefs. Social networks should implement systems to prevent the spread of fake news, however it is not easy to identify them. The main problem is that it is necessary to stop this phenomenon without limiting freedom of expression. Being able to express oneself freely is one of the fundamental values that underlie a healthy democracy. The Russo-Ukrainian conflict is a good example of how hard it is to handle this problem; many social networks stopped spreading news from russian sources, on the other hand, SpaceX refused to block russian news sites to access Starlink with the aim of defending freedom of expression.
Machine learning plays an important role in fighting this phenomenon; thanks to the high computational capacity of modern computers and to the availability of data, in the last years many fake news detection models have emerged. However, this problem proved to be very difficult to solve even for a machine.
In this project, we train different machine learning models to perform fake news classification on a dataset containing US political news.

II Pre-processing of the data

The dataset contains 45000 samples, for each sample we have 4 features: title, text, subject and date. We only take into consideration the first two features dropping the subject and the date since they are not relevant from a linguistic point of view. Moreover, we concatenate the title with the text, therefore, we have only one feature for each news. For each sample we have a label which can assume the value "true" or "fake". The dataset is perfectly balanced. Figure 1 shows a cloud with the most common words in the entire dataset, as we can observe, most of them concern politics.

II.1 Filtering

Filtering is the first step of preprocessing. The goal of this step is to make the dataset more usable by applying filters to the text. First, we remove all the contractions replacing strings such as "isn't" with "is not". Afterwards, we remove all the punctuation. Finally, we also remove stop words like “a”, “the”, “is”, “are”, etc., assuming that they are not important to detect fake news.

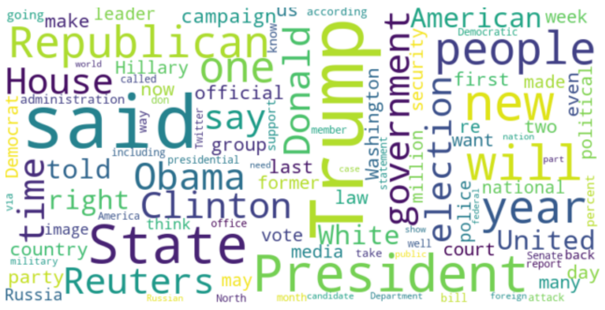

Figure 1: Most frequent words in the dataset

II.2 Stemming

During this phase we are interested in reducing the size of the vocabulary. To achieve this goal we replace the inflected or derived word with their corresponding base form. For example, the words “troubles”, “troubled” are converted to their canonical form “troubl”.
Figure 2 and figure 3 shows the most frequent words for true and fake news after stemming. As we can see, for the two classes the words are very similar. However we notice that the word "Reuter" is present only for the true class.

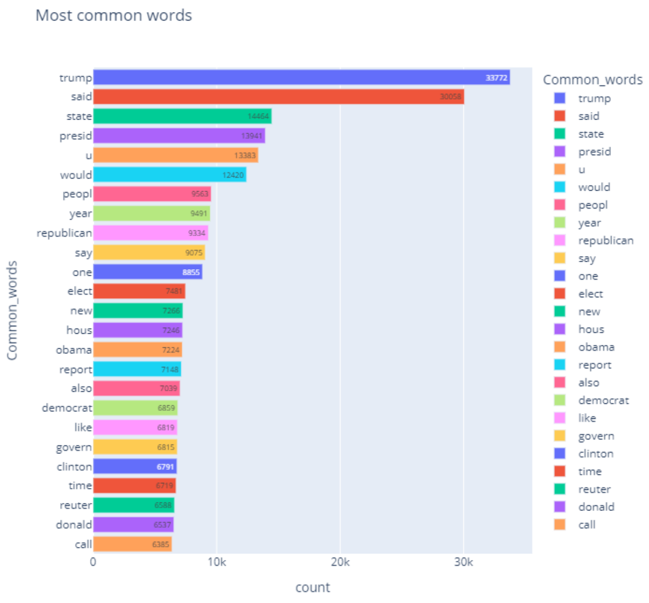

Figure 2: Most frequent words (Fake News)

III Data Embedding

After the preprocessing of the data, the next step is mapping the data to a vector space in order to obtain a representation of the data that is easier to handle. In this project, we use the algorithm word2vec to create an embedding of each word in a text, such embedding will be used to train the LSTM network. Moreover, we use BoW and TF-IDF to create embeddings to train the naive bayes classifier and the logistic regression. Notice that unlike word2vec the last two methods embed the whole text into a single vector. In the following we will give a brief description of these methods.

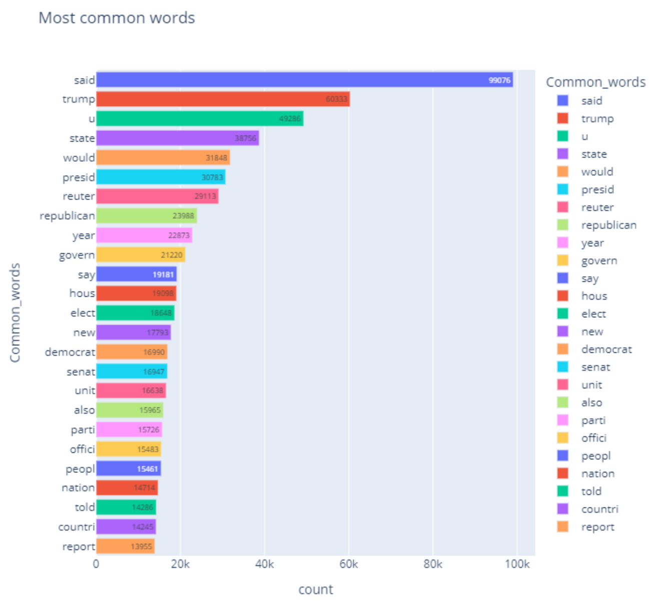

Figure 3: most frequent words on (True News)

III.1 BoW

The Bag-of-Words model is one of the simplest and commonly used to embed a text. Basically, the text is embedded into a vector of size equal to the number of unique words in the document. The value of each component of the vector will be equal to the frequency of that word in the text. Notice that with this embedding we do not take into consideration the position of the words in the text. Therefore two sentences with the same words but different meaning will be represented with same vector.

III.2 TF-IDF

Also in this method the text is embedded into a vector with size equal to the number of unique words. The only difference is that in this case each of the frequency of the words (TF) is multiplied by another term called IDF (inverse document frequency) which consider how frequently such word appear in general in that language. More precisely, for a word x_i in a document with all words $X = \{x_1, x_2, \cdots, x_i, \cdots, x_n\}$, suppose N is the function that gives the number of times x_i appears in the document, the TF is given by

$$TF(x_i) = \frac{N(x_i)}{\sum_{k=0}^{n} N(x_k)},$$

and let D represents the total number of texts in a corpus, and $D(x_i)$ represents the total number of texts in the corpus that contain the word x_i. IDF is given by

$$IDF(x_i) = \log\left(\frac{D}{D(x_i)}\right)$$

Then we get the TF-IDF for the word x_i

$$TF - IDF(x_i) = TF(x_i) \cdot IDF(x_i)$$

Using the TF-IDF, we get more accurate weight of each word. But still, the influence of the order is not taken into consideration.

III.3 Word2Vec

Word2vec embeds each word in a text into a different vector. The idea is that words with similar meaning will be represented by similar vectors.[2]. To embed the words we train a neural network to predict the target word given the nearby words (or vice versa). The final embedding of the word is given by some of the parameters of the network. This method is very expensive from a computational point of view compared to the previous ones. Notice that in this poject we trained the word2vec after stemming but before removing the stop words; in this way, we reduced the vocabulary size but did not change the context of each word (More details can be found in the implementation).

IV Training of the models

In this section, we discuss the training of the models. In particular, we start from a more complex model that is LSTM, and afterwards, we discuss two simple models which are the Naive bayes classifier and the logistic regression. The training set includes 80 per cent of the total samples.

IV.1 LSTM

LSTM (long short term memory) is one of the most used type of recurrent neural network; its main advantage compared to vanilla RNN is that it does not suffer the problem of the vanishing gradient. In this project, we use a many-to-one architecture where at each time step we give as input to the network a word of the article and we expect to get the prediction (fake or true) as the output of the network at the last time step. In particular, we give as input to the network the words embedded using word2vec.
The neural network uses the Relu activaction function, however, the last layer implements a sigmoid function so that we can get a probability for the two classes as output. We also use dropout to improve the generalization capabilities. Notice that to train the network the library Keras requires the input samples (the articles) to be all the same size. We set the size of each sequence to 10000 words since most of the articles are shorter than this number; we truncated all the longer sequences and zero padded the shorter ones.

IV.2 Naive Bayes and Logistic Regression

We decided to use Naive Bayes and Logistic Regression to understand if we could get good results using simple models.
Naive Bayes classifier is one of the simplest techniques to perform classification. To classify a sample, it maximizes the probability of a label given that sample with the naive assumption that the features of the data are independent.
Logistic regression instead, is a model used to perform linear classification on a dataset containing only two labels. The idea is to fit a logistic function minimizing the cross-entropy of the data using gradient descent.
We tried these two methods with both Bow and TF-IDF. The training time was much shorter.

V Results

To test the models we conducted experiments on the test set. We take into consideration two measures: F1-score and accuracy. There are just two possible conditions that could appears in the results. We denote the correct prediction with positive label as TP, and TN for negative label; in the same way, we denote the wrong prediction with positive label as FP, and FN with negative label. Then we have:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2Precision \cdot Recall}{Percision + Recall}$$

All the results are shown in the following tables. All the models worked very well, in fact all accuracies are higher than 0.94. However Logistic regression and LSTM performed much better than the naive bayes classifier. We can see that the results vary significantly when combining different word embedding methods with different classifiers. The BoW embedding outperforms TF-IDF embedding with both Naive Bayes and Logistic Regression. Although LSTM is the most recent and most complex among those models, there is not a significant difference between LSTM and logistic regression, this is becuase the accuracy that we get with logistic regression is extremly high therefore it is hard to get a better model.

	F1-Score		Accuracy	
Naive Bayes	BoW	TF-IDF	BoW	TF-IDF
	0.940	0.933	0.940	0.933

Table 1: Different word embedding with Naive Bayes classifier

	F1-Score		Accuracy	
Logistic Regression	BoW	TF-IDF	BoW	TF-IDF
	0.991	0.987	0.992	0.987

Table 2: Different word embedding with Logistic Regression classifier

	F1-Score	Accuracy
LSTM	Word2Vec	
	0.9964	0.9963

Table 3: LSTM with Word2Vec

VI Discussion and Conclusion

The problem of fake news classification in general is considered hard to solve. It is strange that we could get such good results using the naive Bayes classifier and the logistic regression model. Why do we get such high accuracy? The answer to this question is hidden within the data. Analysing the dataset it is easy to see that the data is biased. In fact, all the true news come from the same newspaper (named Reuter), moreover, none of the fake news come from Reuter. Basically, the dataset assumes that there is a websites that always says the truth, and others that always lie. This assumption is obviously wrong. We can also notice that all the truth news contains the name of the journal (Reuters), therefore it becomes very easy for the classifier to understand if a news is true. Notice however, that also after removing such word the accuracies do not decrease. In fact, there are many other sources of bias in the data. For instance, most of the fake news contains a multitude of citations from Twitter (figure 4) and many more misspelled words compared to the true news (figure 5). This is because the source of the fake news are completely unreliable and their article are low quality compared to Reuters' articles. It is also worth to notice that during the preprocessing of the data we dropped the

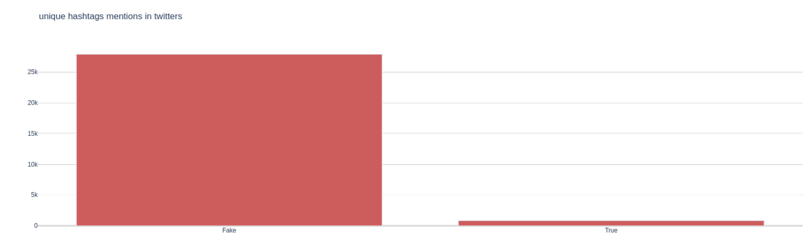

Figure 4: Number of Twitter citations inside the dataset for each class [1]

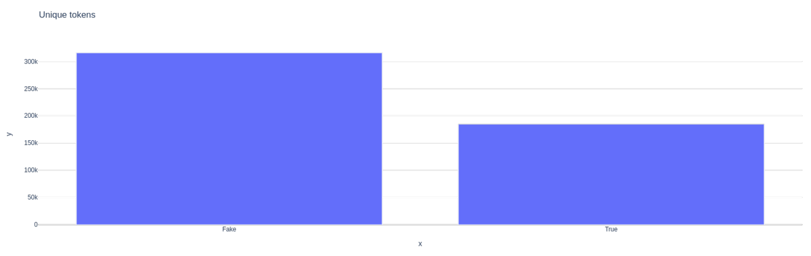

Figure 5: Number of unique tokens for each class. An higher number of unique tokens suggests more misspelled words. [1]

feature named "subject", however, the values that this feature assumes for the fake news do not overlap with values that it assumes for the true news. This basically means that it would be possible to get 100 per cent accuracy just looking at this feature.
To conclude, we obtained very good models of the data, however, our dataset is not representative of the real world, therefore, our models are probably not very useful and they will output correct predictions only in very simple cases.

References

[1] only one word 99.2 per cent accuracy. `https://www.kaggle.com/code/josutk/only-one-word-99-2/notebook`.

[2] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space, 2013.