Fake news detection for Romanian

Parfene Ioana

April 21, 2021

1 Introduction

This article documents the process of using several classifiers in order to identify fake news from Romanian news web sites. The algorithms are tested on different vectorization procedures, lengths of tokenized vocabularies and test data sizes. The sought after information pertains to the accuracy and time of each classification run.

1.1 Motivation

Fake news as a term is defined as the deliberate spread of misinformation via traditional news media or via social media. Its purpose relates to manipulating the masses for political, social, economical or cultural purposes, creating panic or "trolling" as a way of satisfying one's ego or personal view of fighting against a particular moral concept present in current societal norms. It can start with satirical or sarcastic intentions, but snowball and spread at an out of control rate, slowly changing and adapting each time it is shared. Regardless of origin and intent, fake news can rapidly cause damage on a high scale and lasting repercussions. This project aims to take the concept of fake news detection, which is widely known and researched, and apply it to the Romanian language, which is a far less explored territory.

2 Method

The process makes use of Romanian fake and true news articles scraped from real and satire news sites. The text is preprocessed through tokenization and stemming, vectorized and passed through several classifiers in order to find the one that yields the best accuracy and/or time. The detection method was researched through multiple articles and the chosen programming language is Python. As soon as these details were set, many articles were read, a lot of research was done, many tasks were created and a lot of steps were collected into a diagram.

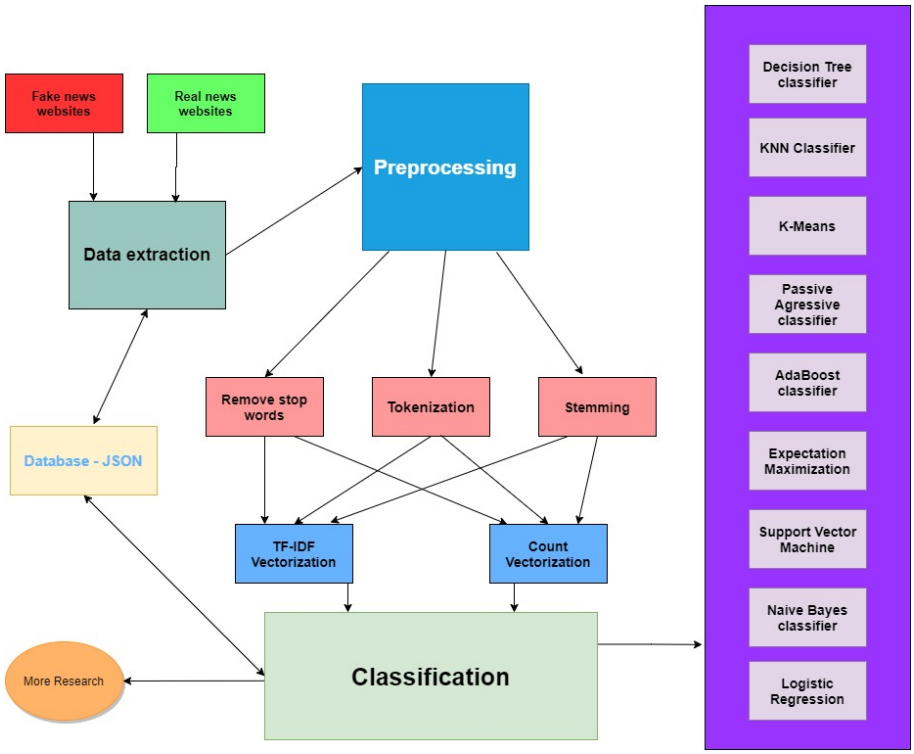

Figure 1: Project Architecture

2.1 Gathering Data

Finding sites that could be crawled and scraped was quite a daunting task. Real news are easy to find, there are countless organizations that have been existing for decades, in television, radio or newspaper formats. Fake news was, on the other hand, tricky. Nobody will write a bunch of false articles, admit to them being fake, then collect them together in a nice place to be found. True malicious or ill-intended posts were impossible to gather, so the next best thing was, logically, satire. Sarcastic news sites are a fraction of the news site pool, especially in Romania. This narrowed down the amount of data that could be collected, as the true corpus size had to be the same as the fake one.

There was also the question of how this restriction would apply into the real world. We used sarcastic articles and we tested them on more sarcastic articles. This can be very helpful against the rumor snowball effect that can lead to the creation of fake news, as well as the tone recognition problem faced by many people. Carefully crafted texts meant to destroy nations, create panic, mimic official and formal articles or exude authenticity would perform worse under this particular data set, but not go completely undetected. The topics chosen by the satire authors and their writings are more times than not meant to poke fun at pieces that circulate as fake news. They have their fair share of "true fake news" characteristics.

2.1.1 Web Sites

The final corpus has 200 000 words, used across 1200 articles, equally spread between the two categories. There were 8 sites crawled and scraped, half for true news, half for satire.

Category		**True**	**News**			**Fake**	**News**	
Web Sites	**ProTV**	**Digi24**	**Libertatea**	**Realitatea**	**TNR**	**Cațavencii**	**7lucruri**	**TimpuriGrele**
Articles	124	89	91	108	42	156	502	96
Words	41,796	69,800	41,981	46,508	7,917	98,544	055,899	37,690

Figure 2: **Web Site Corpus Distribution**

2.1.2 Crawling

The article link gathering was done through a process called "crawling", which means recursively finding and saving all the links found on a certain web page. Those saved links are used to continue the search, until no more are found or until a set limit. Some websites were tough to crawl, as they only advertised recent articles. This lead to a disproportionate distribution of total words obtained across all websites, but he main goal of finding equal amounts of true and fake news was met. The implementation was relatively easy, done using Python libraries.

2.1.3 Scraping

The actual downloading of the article text is called scraping. It involves using the HTML tags of the web sites in order to locate the content, which was quick as all the news sources fit into a pattern and only required one function with different inputs to scrape.

2.2 Preprocessing

The articles had to be preprocessed before being passed through the classifiers. The first step was combining the title and content of each piece into one string. The second step was removing the stop words, then turning the string into individual words. The last process chosen in this particular implementation was stemming, which means reducing words to a short root that represents the differently-spelled versions of a certain notion.

2.2.1 Stop Words

2.2.2 Tokenization

2.2.3 Stemming

2.3 Data Models

2.3.1 Bag of Words

2.3.2 TF-IDF

2.4 Classifiers

2.4.1 Naive Bayes

2.4.2 Passive Aggressive Classifier

2.4.3 Logistic Regression

Words		**200**				**1000**				**19518**		
Model	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**
Test	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**
Result	82.5%	82.9%	82.6%	82.9%	90.3%	89.7%	87.4%	86.6%	93.3%	93.3%	73.0%	71.8%
Time	0.01s	0.01s	0.01s	0.01s	0.09s	0.08s	0.05s	0.06s	1.76s	1.67s	0.98s	1.08s

Figure 3: **Naive Bayes**, Accuracy Table, **Average** out of **100** runs

Words		**200**				**1000**				**19518**		
Model	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**
Test	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**
Result	83.0%	83.4%	84.3%	84.5%	93.1%	93.3%	93.5%	92.6%	94.3%	93.8%	94.2%	94.0%
Time	0.03s	0.03s	0.03s	0.03s	0.13s	0.13s	0.17s	0.15s	2.50s	2.3s	2.79s	2.63s

Figure 4: **Passive Aggressive Classifier**, Accuracy Table, **Average** out of **100** runs

Words		**200**				**1000**				**19518**		
Model	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**
Test	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**
Result	86.1%	85.9 %	85.5%	84.8%	93.8%	93.2%	84.2%	82.9%	93.7%	93.2 %	72.1 %	71.4%
Time	0.10s	0.09s	0.04s	0.04s	0.25s	0.22s	0.13s	0.11s	3.98s	3.84s	1.86s	2.14s

Figure 5: **Logistic Regression**, Accuracy Table, **Average** out of **100** runs

Words		**200**				**1000**				**19518**		
Model	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**
Test	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**
Result	75.7%	75.3%	73.7%	73.4%	75.0%	74.2%	73.3%	73.0%	71.6%	71.1%	67.8%	66.2%
Time	0.02s	0.03s	0.01s	0.02s	0.09s	0.09s	0.06s	0.06s	1.69s	1.71s	1.03s	1.09s

Figure 6: **K Nearest Neighbour(2 neighbours)**, Accuracy Table, **Average** out of **100** runs

2.4.4 K Nearest Neighbour

2.4.5 Support-vector machine

Words		**200**				**1000**				**19518**		
Model	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**
Test	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**
Result	86.1%	85.9 %	75.5%	74.8%	90.1%	89.2%	81.2%	79.9%	92.7%	91.2 %	90.1 %	89.4%
Time	0.32s	0.33s	0.31s	0.32s	1.09s	1.09s	1.06s	1.06s	20.69s	20.71s	15.03s	16.09s

Figure 7: **SVM**, Accuracy Table, **Average** out of **100** runs

2.4.6 Decision Tree Classifier

Words		**200**				**1000**				**19518**		
Model	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**
Test	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**
Result	89.5%	88.4%	87.2%	87.7%	88.2%	88.3%	87.1%	86.7%	80.5%	80.1%	80.8%	80.1%
Time	11.8s	9.5s	9.9s	7.0s	0.23s	0.19s	0.26s	0.22s	0.05s	0.05s	0.06s	0.05s

Figure 8: **Decision Tree Classifier**, Accuracy Table, **Average** out of **100** runs

2.4.7 Ada Boost Classifier

Words		**200**				**1000**				**19518**		
Model	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**	**BoW**	**BoW**	**T-I**	**T-I**
Test	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**
Result	93.4%	93.2%	92.2%	92.9%	93.5%	93.1%	92.8%	92.4%	86.3%	86.8%	86.2%	86.1%
Time	31.9s	28.5s	32.4s	29.3s	1.3s	1.1s	1.5s	1.3s	0.2s	0.2s	0.3s	0.3s

Figure 9: **Ada Boost Classifier**, Accuracy Table, **Average** out of **100** runs

3 Results

4 Conclusion

References