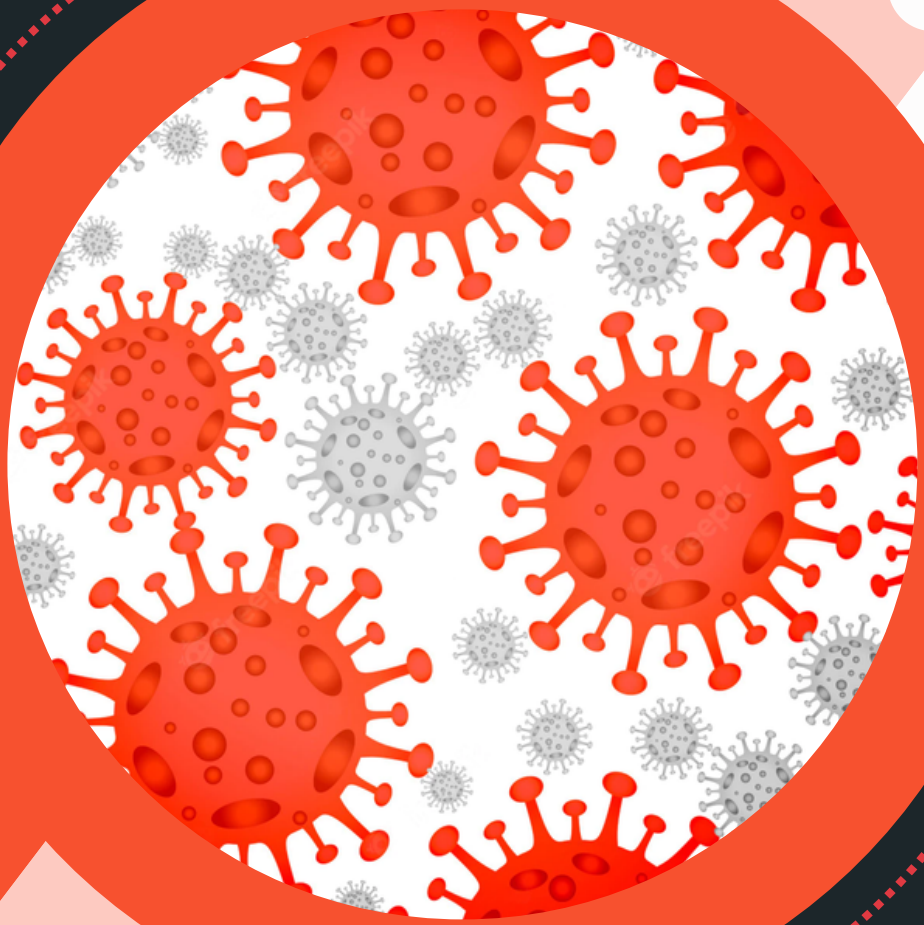
COVID FAKE NEWS DETECTION

Text Analytics Project
2022/2023



Davide Di Virgilio
Chiara Menchetti
Bruno Limon
Carlo Volpe

Contents

1 Introduction

Misinformation has always been present in society, however with the advent of technology its spread has been uncontrollable. As the world was facing the wake of the COVID-19 pandemic in 2020, fake news increased dramatically. This resulting in an overall panic in the population of even bigger proportions than it should have been. Fake news coming from the web and social media are, especially for people with lower levels of digital awareness, dangerous, and could create a distorted vision of reality. The study proposed in this project is the analysis and detection of such inaccurate information regarding the Coronavirus pandemic.
The data was gathered from the COVID-19 Fake News Detection repository from Kaggle, which contains a collection of 33 files made up of articles, tweets and claims.

2 Data Understanding and Pre-processing

The focus of the analysis will be on the 13 files containing only articles and tweets, disregarding the ones with claims and retweets. The features of the data depend on the file. For the articles there are more features, namely title, text, url and others, whereas for the tweets there is only the tweet id. The idea would be to concatenate all of the data together and assign to each record whether it is real or fake.
The first step was to load the files separately and create a new variable *fake*, which can be either 1 (the observation is fake) or 0 (the observation is not fake). A further task was necessary for the files containing tweets, since only an id was available. Their contents were fetched using the library *Tweetpy*, which connects to Twitter's API through the keys and tokens provided with Twitter's developer account and is able to request the contents associated with a certain id. With this method, a new corpus containing tweet news and their label was then created.

Given the intent of having the two forms of news together, but having different variables, the number of these was reduced. The combined dataset has 129880 records across only 2 features: the *content* which corresponds to the headline of the article or text of the tweet, and the label *fake*, which indicates their correctness.
The next phase of the pre-processing is to clean the textual part of the dataset by converting it to lower case letters and removing all the unnecessary characters (for the analysis of this project) such as links, mentions and emojis. It may happen that data contains duplicates, and since it was not collected by us, it is standard procedure to check the *content* variable. After the drop of all the duplicate rows, the total number of observations left was 85,350. After this reduction, the data was checked manually to overcome unnecessary surprises. However, it was found out that in the dataset at this point there were still some records having almost equal content, and differed by just a word or an additional character. There is no information on the criterion chosen by the authors for gathering this data, however, since the idea would be to have as a diverse data as possible, the presence of such, even partially, equal observations, would constitute bias. An example of this in Table 1.

	Content	Fake
129413	Bill Gates Explains that the Covid Vaccine wil...	1
129414	Bill Gates Explains that the Covid Vaccine wil...	1
129415	Bill Gates Explains that the Covid Vaccine wil...	1
129416	Bill Gates Explains that the Covid Vaccine wil...	1
129417	Bill Gates Explains that the Covid Vaccine wil...	1

Table 1: Duplicate rows

To tackle this issue all those records having the first 30 characters equal to some other record were considered duplicates and removed from the dataframe. To clarify, this might not be a problem (to have some almost identical records), however to make the classification task more interesting it was decided to proceed in this direction. At this point, the size of the dataset had shrunk some more, reaching 66,681 rows, which is unfortunately half of the initial raw one, but at least the data is clean and without any duplicates.
Moving on to the data understanding, the first aspect to check is the proportion of the fake and real news.

The number of real news is 64,353, whereas the number of fake news is 2,328. The dataset is highly unbalanced, however given the task of fake news detection it is quite normal to have these kind of proportions between the two classes, this aspect will be taken into account when performing any supervised classification algorithm.
The other variable to explore is the *content*. Given its nature, the one aspect that can be checked is the frequency of the words. The following two representations (Figure 1) are wordclouds, showing the most frequent words in the data. In order to perform the frequency of the words, tokenization must be performed and stopwords must be removed. For the Figure 1(b), to the list of stopwords removed were added the words coronavirus, covid and covid19, which represent the topic of the data, so not really meaningful.

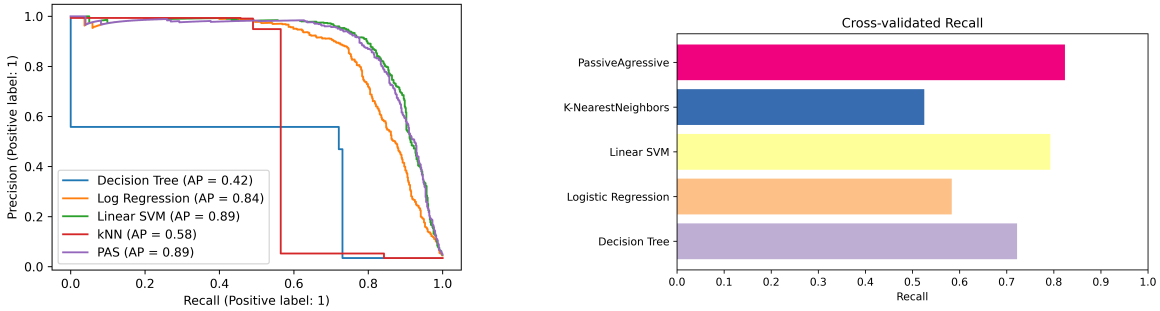
Figure 1: Wordclouds on the cleaned dataset

The most frequent words, except from Coronavirus or Covid, are *people, new, health, amp, resource, testing, home, help, case*. If you had the opportunity to live through this last few years these are probably some of the words we have heard over and over again, and that could have effectively described this period in time. From what it was possible to gather from the data, it dates back to 2020/2021, and it regards probably mainly US news. A word that might not be familiar to everybody is AMP (Analysis and Mapping of Policies) which is a comprehensive database of policies and plans to address the COVID-19 pandemic.

3 Classification

Regarding the classification task, the first thing to do is to prepare the data to be fed into the classifying models. To do this, it was necessary to apply Word Embedding, a technique that allows to represent the raw documents, in this case the *content* attribute, into a vector of real values within a predefined vector space, essentially, each word is mapped to one vector and the vector values are learned in a way that resembles a neural network. This approach leads to a good performance due to the type of distributed representation, based on the usage of words, allowing words that are used similarly, to have similar representations, something preferred over other methods such as one-hot encoding, where the result is a much bigger sparse representation, or a Bag of Words approach, where, unless explicitly managed, words have different representations regardless of how they are used.
To apply these kind of algorithms, the main options available are *Word2Vec*, *CountVectorizer* and *TF-IDFVectorizer*. After a quick implementation of simple classification models using these word embedding methods to prepare the data, TF-IDF is the chosen method due to a slightly better performance and fastest computational times.

Having the data ready and in a suitable vectorized form, it is now possible to see how some of the most common classifying models perform on the task while using the default parameters. To do this, 5 models are trained and fitted into a 10-fold cross validation model, measuring *accuracy*, *precision*, *recall* and *f1-score*. The results are decent and fairly similar for all models across most metrics, except for *recall*, where some models perform very poorly, which confirms the highly unbalanced nature of the dataset, and thus, a high number of false negatives appear, implying the model has a hard time identifying some of the fake news, classifying them instead as real, something undesirable in this context. An overview of these results can be seen in figure 2, showing the precision-recall curve for each model, as well as their recall score. Both fig. 2(a) and 2(b) show how the *decision tree* and *k-nearest neighbor* classifiers are the worst performing.



(a) (b)

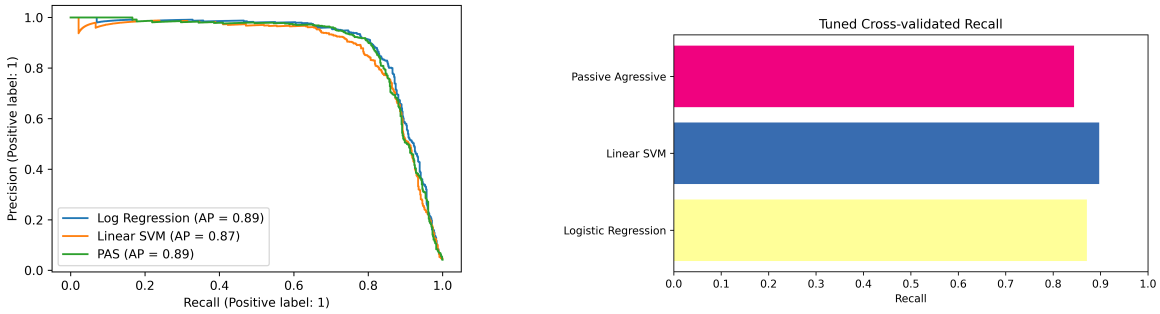
Figure 2: Cross-validation results

To solve this, several techniques can be implemented. First, an attempt to rebalance the dataset, using *RandomUnderSampling*, which removes random samples of the majority class, in order to have a more balanced proportion, using a ratio of 1/10, it is possible to transform the original proportion of Real:45,046/Fake:1,630 to Real:16,300/Fake:1,630. With this more balanced data, the recall metric slightly increases, but only by about 2% in most models, which indicates further attempts must be performed to reduce the number of false negatives.

The next method to try is to perform hypertuning of the parameters of the classifying models, to begin with it, a random search was implemented to choose random combinations of parameters from a grid, fitting the model with the same cross-validation as before and returning the model with the highest score obtained, in this case, with the highest recall, with the objective of reducing the amount of false negatives. Having obtained these models, the worst performing algorithms from before saw little to no improvement with the new parameters, so they are discarded and from now on the focus relies only on *Linear Support Vector Machines*, *Passive Aggressive* and *Logistic Regression*. Now a more thorough search can be performed, using gridsearch with a finer grain between the parameters, this also returns the highest scoring (recall) models. The final models to test after trying some of the combinations of parameters are:

- LogReg(C=110, class_weight='balanced', fit_intercept=False, penalty='l2', dual=True, solver='liblinear')
- PassiveAggressiveClassifier(C=0.25, class_weight='balanced', fit_intercept=False)
- LinearSVC(C=0.2, class_weight='balanced', fit_intercept=False, loss='hinge')

These models are again evaluated using cross-validation, whose results can be seen in figure 3, showing the same information as in figure 2, but after tuning and obtaining the best possible performance. In addition to this, it is possible to see all results in table 2, showing performance before and after tuning for *Logistic Regression* LR, *Linear Support Vector Machines* SVM,and *Passive Aggressive Classifier* PAS.

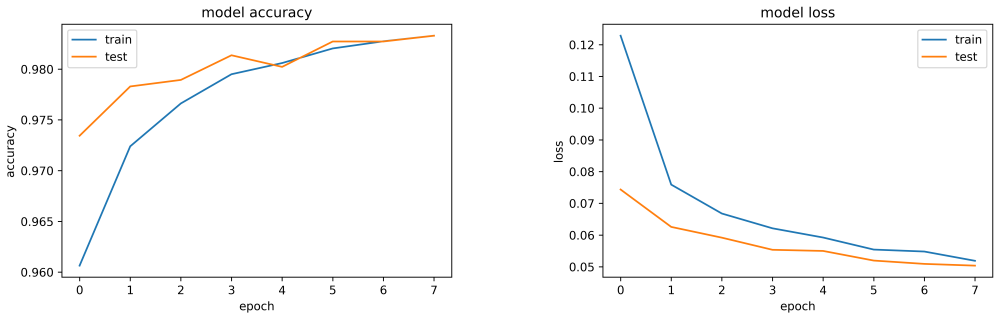


(a) (b)

Figure 3: Cross-validation results after tuning

3.1 BERT

Bidirectional Encoder Representations from Transformers is another promising tool to use in this task. Developed by Google, BERT is a pre-trained language model with a state-of-the-art performance for text classification. In contrast to other models, the transformer on which BERT is based uses an encoder that reads the documents in a bidirectional way and thus is better able to understand the context of a word and its surroundings, both left and right. To feed a dataset to BERT, it is first necessary to further preprocess it using its own encoder, *BertTokenizer*, which transforms the documents into a tensorflow tensor called *attention_mask* and their corresponding *input_id*, these 2 arrays will form the first input layers that make up the structure of the deep learning neural network that uses BERT. The next step is to build the model with an adequate optimizer and callback function, for this task, the specific pre-trained model to use was *bert-base-uncased*, after this, the training of the model can begin. It is worth noting though, that this model is very computationally expensive, this due to the size of the model, which has more than 100 million parameters to train, as such, the number of epochs that can be performed is limited by the available system resources. For this task, 8 epochs were decided. After the training phase, scoring and loss results can be seen in figure 4.



(a) (b)

Figure 4: BERT model results

After building and training the model, it is ready to be used to classify the observations in the test set. Its results are good but not better than the previously fine-tuned models, scoring high in accuracy as the rest of the models before but lower on precision and recall, confirming the ongoing trend of difficulties to predict the minority class and resulting in higher numbers of false positives/negatives, most likely due to the unbalancedness of the data, as well as the limited resources that did not allow to have a thorough analysis beyond a brief overview of its functioning.

Final models cross-validated performance					
		Accuracy	Precision	Recall	F1-Score
LR	Baseline	.9770	.9954	.5204	.6826
	Tuned	.9874	.8770	.8558	.8664
	Improvement	1.06%	-11.89%	64.45%	28.96%
SVM	Baseline	.9876	.9758	.7583	.8531
	Tuned	.9812	.7568	.8937	.8190
	Improvement	-0.64%	-22.44%	17.85%	-3.99%
PAS	Baseline	.9883	.9362	.8099	.8680
	Tuned	.9874	.8943	.8339	.8627
	Improvement	-0.09%	-4.47%	2.96%	-0.61%
BERT	Baseline	.98	.88	.85	.86

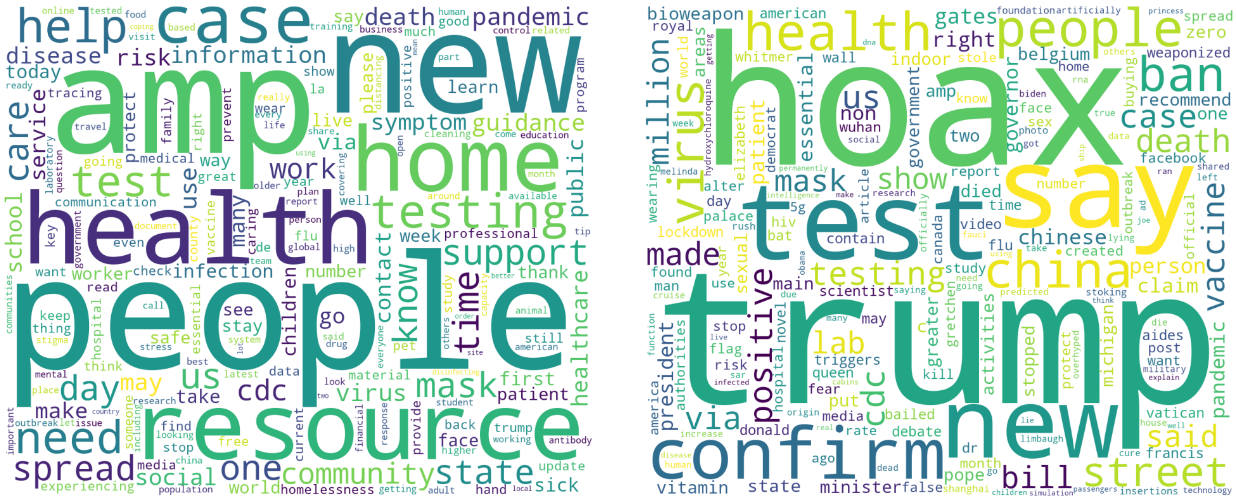
Table 2

From the results seen in Table 2, all three of the first models benefited from the 'class_weight = balanced' parameter, assigning class weights as $n\_samples/(n\_classes * np.bincount(y))$, which is very useful for this heavily unbalanced data. Furthermore, a clear trade-off between precision and recall can be appreciated, this due to the balance between false positives and negatives, as such, a decision between the approach to take must be made. In the case of fake news detection, this analysis considers false negatives to be more costly, since that would mean that a fake news article is passed as real and can therefore cause more damage than if a real news is wrongly deemed as fake. The model that decreases false negatives the most is *SVM*, but suffering in precision, while a more cautious approach might be found in Logistic Regression, with a more balanced performance. Regarding BERT, its state-of-the-art performance is apparent straight out of the box, with the best results overall (although just slightly), on the cross-validated evaluation, however, this is achieved without any further tuning, using just the pre-trained model as reference, hinting at the possibility of much better results should the computational results be adequate for further testing.

4 Sentiment Analysis

Once the fake detection phase was over, the work continued with the aim of understanding the target sentiment of the two classes. The methods used were Sentiment and Emotion Analysis.
The work was started on the dataset without duplicates, a new first phase of text cleaning, in which stopwords, links, https etc. were removed, was followed by one in which the dataset was divided by filtering for fake (1) and real (0). The idea would be to analyse the fake and real observations separately and then compare them. To get a first glimpse of the most frequent terms for the two splits, the wordclouds in Figure 5 were constructed.

Subsequently, an unsupervised sentiment analysis was performed on both subdatasets (Fake and Real) using the Vader_Lexicon dictionary, which contains words already labelled as positive, negative and neutral. Specifically, the first step was the stemming and tokenization of the words, then the sentiment score was calculated using the dictionary: once each word of the text was labelled, an overall sentiment score was derived by counting the number of positive and negative words and mathematically combining these values (number of positive words - number of negative words / total number of words). The sentiment scores, positive values, negative values and zero, automatically indicate the positive, negative or neutral impression of the text. Looking at the frequencies of the sentiment score, Negative (1102), positive (595) and neutral (631) present in the dataframe of Fake news it can be noticed how the negatives are almost twice as high as the positives, whereas in the dataframe of Real news the positives (31616) are twice the negatives (16324) and the neutrals (16413). This gives us an insight into which sentiments the Fakes and the Reals are pointing at.



(a) Real news (b) Fake news

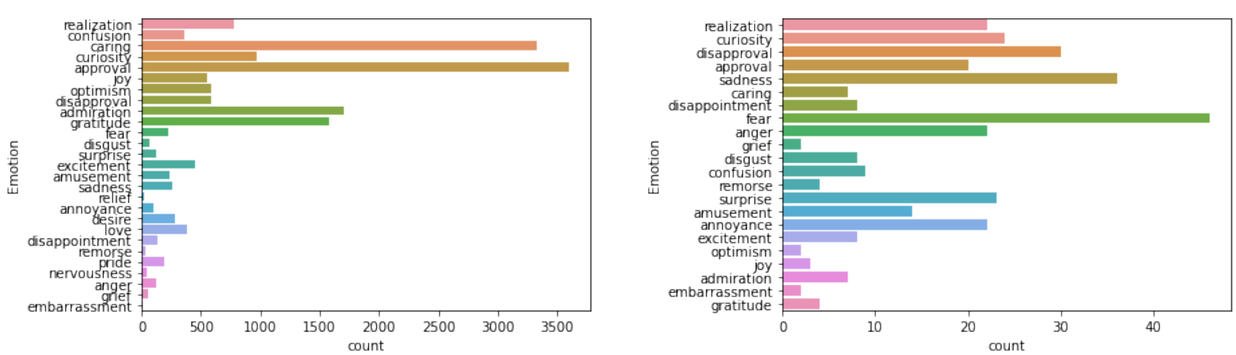
Figure 5: Wordclouds

4.1 Emotion Analysis

At this point, an Emotion Analysis was carried out on the results produced by the previous Sentiment Analysis, i.e. on FN Negative - FN Positive - FN Neutral and on RN Negative - RN Positive - RN Neutral. Unlike Sentiment Analysis which classifies texts into positive, negative and neutral classes, Emotion Analysis consists of "multiclassifying" the text into different types of emotions based on what the text conveyed. With this technique, it is possible to map opinions in a multidimensional space of emotions that allows us to guess which emotional levers are used for Fake and Real News.
In particular, two separate algorithms were used, to classify the emotions into:

- fear - anger - trust - surprise - positive - negative - sadness - disgust - joy - anticipation (by *NRCLex* function algorithm);
- confusion - approval - neutral - curiosity - caring - realisation - anger - optimism - disappointment - surprise - disapproval - joy - pride - disgust - admiration - sadness - boredom - desire - amusement - fear - nervousness (by algorithm using the *EmoRoBERTa* model).

Starting from the results of the Sentiment Analysis which show us that Fake dataframes point to negative sentiment and Real dataframes point to positive sentiment, we can verify the emotions that both Fake and Real produce.



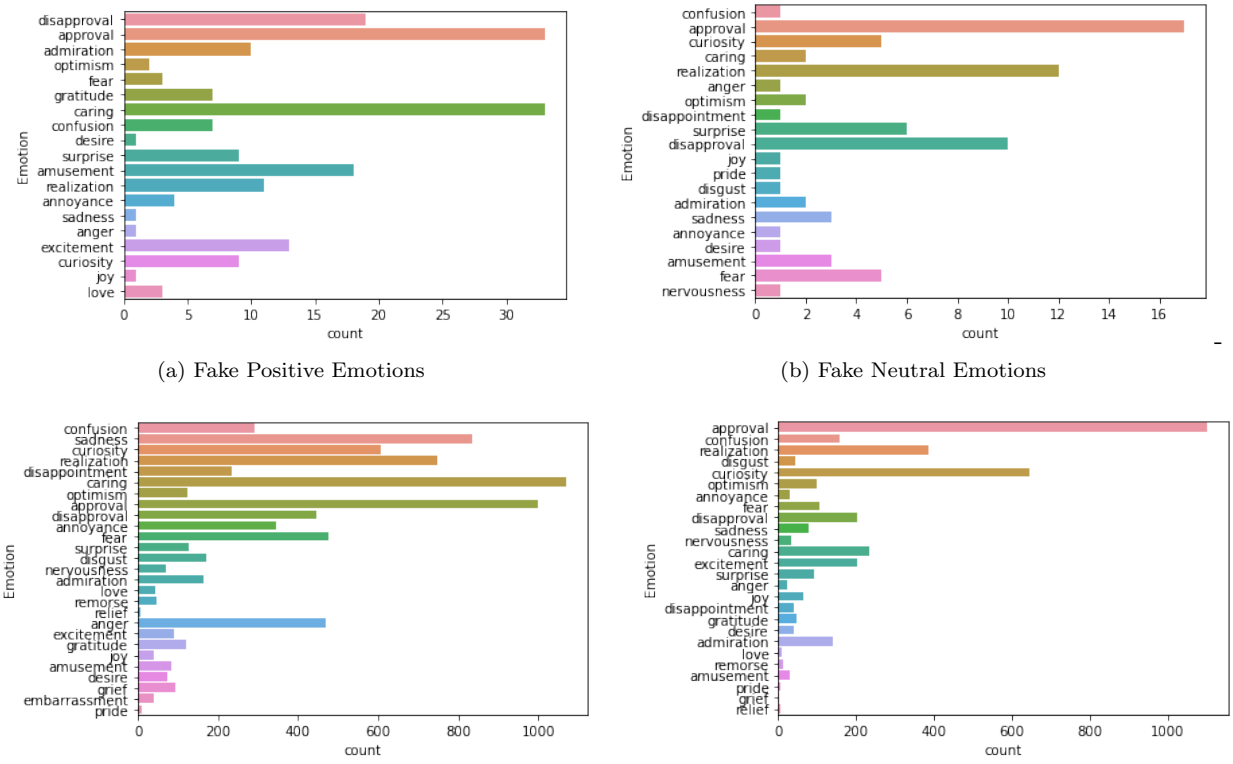
(a) Real Positive Emotions (b) Fake Negative Emotions

Figure 6: Emotions from the algorithm based on the *EmoRoBERTa* model

	Average Emotions Score - NRCLex									
	Fear	**Anger**	**Trust**	**Surprise**	**Positive**	**Negative**	**Sadness**	**Disgust**	**Joy**	**Anticipation**
DF Fake Negative	0.0895	0.0628	0.1016	0.0816	0.1879	0.8903	0.0817	0.0585	0.0208	0.1469
DF Real Positive	0.0645	0.0258	0.1722	0.0305	0.3258	0.0975	0.0449	0.0198	0.0783	0.1796

Table 3: Average Emotions from the *NRCLex* function

Below are the overall results of the emotional analysis on the different Sentiments:



(a) Fake Positive Emotions

(b) Fake Neutral Emotions

(c) Real Negative Emotions

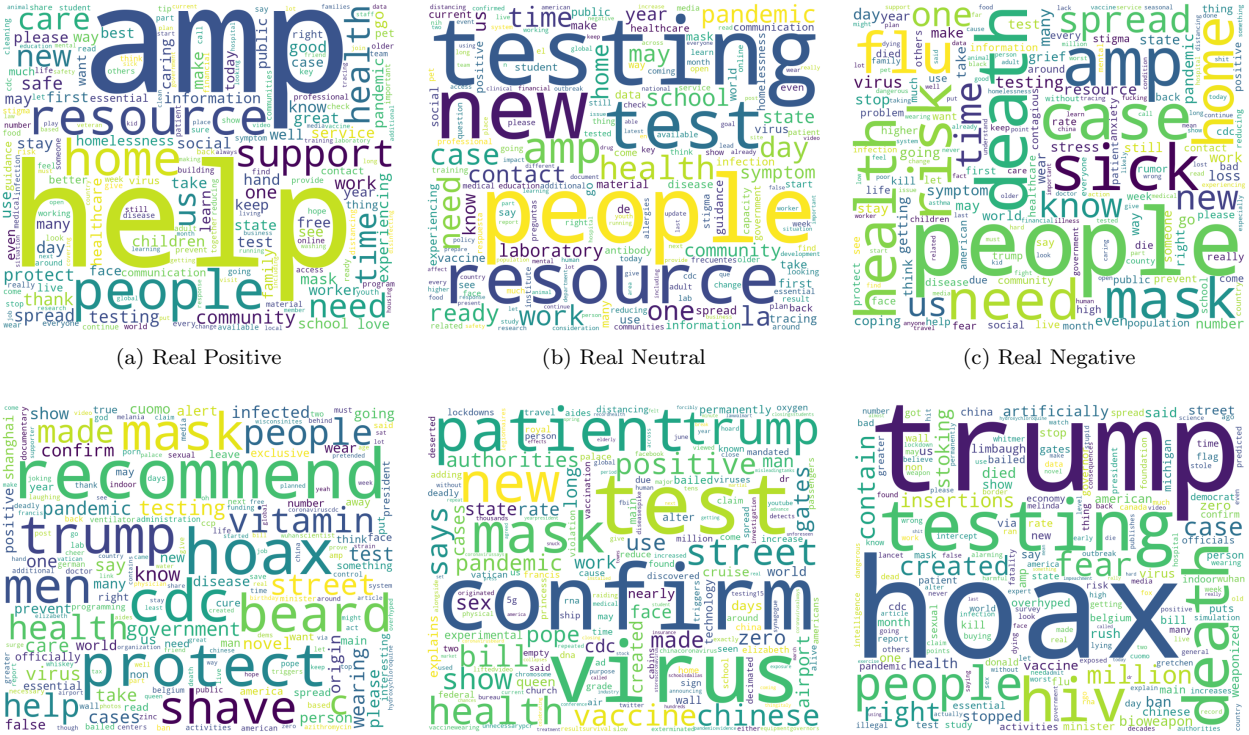
(d) Real Neutral Emotions

Figure 7: Emotions from the algorithm based on the *EmoRoBERTa* model

	Average Emotions Score - NRCLex									
	Fear	**Anger**	**Trust**	**Surprise**	**Positive**	**Negative**	**Sadness**	**Disgust**	**Joy**	**Anticipation**
DF Fake Positive	0.0860	0.0304	0.1326	0.0543	0.2671	0.1632	0.0459	0.0398	0.0487	0.1730
DF Fake Neutral	0.0535	0.0168	0.1233	0.1340	0.3141	0.1069	0.0346	0.0151	0.0328	0.2855
DF Real Negative	0.1281	0.0595	0.1100	0.0311	0.1753	0.2124	0.0935	0.0539	0.0289	0.1478
DF Real Neutral	0.0647	0.0215	0.1484	0.0247	0.2777	0.1007	0.0423	0.0184	0.0313	0.2672

Table 4: Average Emotions from the *NRCLex* function

For each analysis, a wordcloud was produced to highlight the most frequent terms by removing, according to the results of the second algorithm, the observations with a neutral value so that the most significant terms could be highlighted.



(a) Real Positive (b) Real Neutral (c) Real Negative

(d) Fake Positive (e) Fake Neutral (f) Fake Negative

Figure 8: Wordcloud on the six different datasets

As a concluding remark, the work approaches an intent analysis on the basis of the results of the Emotion Analysis, which could become predefined classes or reference intents. By adopting this approach, it is possible to observe in particular the incidence of the fake news on the emotions during the Covid-19 pandemic. Certain topics correspond to certain emotional reactions. In particular, we observe how fake news exploit negative emotions as a media factor.

5 Topic Modeling

Finally, Topic Modeling was performed, which is useful for determining the topic of a set of documents based on content. The observations were divided according to the target variable with the goal of evaluating how real and fake news can deal with the pandemic theme while involving different scenarios. The first step was text cleaning, followed by vectorizing the observations in order to apply them to the *LatentDirichletAllocation* model. The parameters chosen were the following:

- Max_iter = 20
- Learning_method = 'batch'
- Verbose = 1
- N_jobs = -1

The optimal split identified for real news is 4 topics, 3 for fake news. These numbers were obtained from an empirical observation done on the plot generated by *pyLDAvis*, which allows to interpret how the text corpus has been fitted into a particular topic. The top 10 words by frequency of each topic are shown below:

		Top 10 Words
Fake	Topic 0	hoax, trump, street, testing, cases, right, said, us, stopped, michigan
	Topic 1	says, bill, vaccine, gates, tests, via, million, positive, deaths, trump
	Topic 2	made, health, lab, essential, ban, non, two, belgium, minister, non essential
Real	Topic 3	guidance, information, cdc, new, resources, disease, use, media, via, study
	Topic 4	testing, cases, health, test, new, contact, deaths, symptoms, infection, tracing
	Topic 5	people, home, get, like, know, go, sick, one, need, amp
	Topic 6	amp, support, health, resources, people, community, healticare, care, help, homelessness

Table 5

In the analysis of scenario identification, it must be taken into account that a word may occur in more than one cluster. Therefore, to determine which words will form a topic, it was decided to consider only words with a frequency greater than 60% of the total frequency of that word. In this way, it was possible to determine the topics as shown in the following table:

		Topic scenario
Fake	Topic 0	Trump: His statements on Covid and vaccines
	Topic 1	Covid dissemination: restrictive measures by two women in power
	Topic 2	Viral information: conspiracy and deniers
Real	Topic 3	Health and everything around it: people, patients, doctors, clinics
	Topic 4	Illness and how to cope with it: treatments, masks, behavior
	Topic 5	Communication: news and information
	Topic 6	Monitoring: cases, deaths, propagation

Table 6

This analysis shows how political propaganda plays a central role in the dissemination of pandemic news. While in real news topics, the information revolves around issues such as health care, treatment updates, or the disease itself, in topics derived from fake news it is possible to see news articles related to statements by political figures such as Trump, Queen Elizabeth's containment measures, or Michigan Secretary of State Whitmer Gretchen. Furthermore, the presence of Bill Gates' name confirms how much fake news revolve around.

At the end of this task, it can be seen how fake news are most often linked to public personalities and people in power, this in an attempt to increase their plausibility and reach a greater audience, influencing their opinion, showing the risk and dangers of an uncontrolled spread of fake news.

6 Conclusion

Even if at first the sole objective was to detect fake news, during the analysis, we realized how important it was to go further, trying not only to determine the truthfulness of an article, but also to understand the characteristics on which fake news are based and to discover through topic modeling on which scenarios fake news lean on. As a matter of fact, we have seen how fake news tend to aim to more negative sentiments in an attempt to arouse emotions such as fear, uneasiness, disappointment or curiosity.

A viable further step could be that of analyzing the effect that a particular sentiment and/or topic might have on the rate of spreading that a news article could present. This could be checked by counting the amount of retweets that certain fake news articles have, implying that there is indeed a correlation between the spread of misinformation and the underlying emotions and scenarios that accompany the news.