CentraleSupélec

Projet d'option
Report

Fake news detection

Student :
Abdellah Kaissari
Abderrahim MEHDAOUI

Encadrant :
Gianluca QUERCINI
Tuteur :
Fragkiskos Malliaros

March 30, 2020

Abstract

Social networks are nowadays often used to get informed and read news. Since then, there has been an increasing amount of fake news for various commercial and political purposes. With deceptive words, social network users can be infected by these online fake news easily, which has a very bad effect on both the online and the offline society. The goal of this project is to build an artificial intelligence model to detect fake news automatically and to check the trustworthiness of information on social networks. In fact, we framed this problem as a supervised classification problem and we used the benchmark dataset Liar introduced recently in [1]. This project is divided in two parts, first part we tried different machine learning and deep learning models using different techniques as tf-idf, LIWC, word2vec and Elmo embeddings to detect fake news. In the second part, we sub-divided the dataset with respect to the author and with respect to the political affiliation and we tried detection fake news in theses sub-divided datasets. In this project, we also focused on the effect of adding LIWC to tf-idf and Elmo embeddings. At the end of this project, we obtained a neural network that have a F1 score of **.281** and **.645** on the 6 classes problem and the binary problem respectively in the validation set. These scores are close to state-of-the-art models using the same kind of studies.

Contents

1 Introduction

The detection of fake news is a major concern nowadays. In the era of social networks, a fake news spreads up to six times faster via Twitter than a reliable news item, according to a study carried out by MIT researchers in 2018. This propagation of fake news has a negative effect on social networks users but also on people who do not use them. In general, the surprising and attractive nature of this type of news make its propagation way easier than usual. The field of fake news has also an important impact in its propagation. We often find topics related to politics, terrorism and natural disasters. The spread of fake news can for instance have a great impact on election results or public opinion on a sensitive topic such a climate change or Coronavirus. With the development of Machine Learning, Deep Learning and Natural Language Processing (NLP), researchers have recently focused their efforts on fakes news. Several articles have been published on this field in recent years.

In this project, we have used the Liar dataset [1] presented in details in section 3. It is a dataset containing thousands of political statements, each of them is associated to one of the 6 labels (from 'true' to 'pants-on-fire') that help us to appreciate their veracity. This project is divided into 2 parts, the first part studies the supervised classification problem using the 6 classes presented before and the 2 classes problem ('true', 'false'). The second part studies the binary classification problem on sub-sets of the dataset grouped by the speaker or by the political affiliation. In the two parts, we have focused in the effect of using LIWC [2] on the quality of our models. LIWC is a word counting model that refers to a dictionary of grammatical, psychological, and content word categories. It is a tool used by scientific to take on consideration the psychological aspect of words. Adding the features from LIWC to the features obtained from the text by using other methods (tf-idf, word2vec) might ameliorate the models in many cases. We have used theses tools with logistic regression, Svms, random forest and neural networks. During all the project, we have used the weighted F1-score measure. It considers both the precision and the recall to compute the score. In this studies, it is more adapted to use this measure than the accuracy since the dataset used is not balanced and we do not want to misclasify a fake news.

The report is organized as follows: Section 2 describes the related work about fake news detection, the section 3 presents the dataset used during this project. Section 4 gives a detailed description of LIWC. Section 5 details the general methodology we followed and section 6 presents the obtained results. Finally, section 7 provides the final conclusion.

2 Related work

According to [3], fake news detection approaches can be classified along two axes. The first one uses a **Linguistic Approaches** where the textual content of the news is extracted and analyzed to associate or find particular language patterns which are associated to whether truth or lies. The other axis uses a **network approaches** in which we use network information, such as message, metadata or structured knowledge network queries as measures of veracity. Machine learning method can be incorporated in both approaches to perform the analysis. In this paper, we use a linguistic approach using some linguistic patterns as tf-idf, word embedding or LIWC.

The veracity of a news can be categorized as either true or false (binary categorization) or be assigned to more labels (6 in our project: true', 'mostly-true', 'half-true', 'barely-true', 'false', 'pants-on-fire') since most of news aren't completely true or false . The paper [4] has tried to perform fake news detection on both categorization (with 6 classes and 2 classes) using logistic regression, Naive Bayes and LSTM network. they used the Politifact dataset statements. It is the same statements as in the Liar dataset used in our project. They have first computed the ratio of averages between fake news and true news for some of their measured features including LIWC metrics. Ratios greater than one denote features more prominent to be fake, and ratios less than one denote features more prominent to be true. In their work, they compared between classification using textual features and classification using LIWC vectors. They also combined the two techniques. The tables 1 and 2 summarize their results respectively obtained on their validation and test set for the politifact dataset. Theses tables presented the macro F1 measures. The F measures is more used in this kind of problem where the dataset is not balanced.

Model	2-Class		6-Class	
	text	+LIWC	text	+LIWC
Naive bayes	.44	.58	.16	.21
MaxEnt	.55	.58	.20	.21
LSTM	.58	.57	.21	.22

Table 1: F1-score on the validation set

Model	Feature	2-Class	6-Class
Naive bayes	text + LIWC	.56	.17
MaxEnt	text+ LIWC	.55	.22
LSTM	text+ LIWC	.52	.19
LSTM	text	0.56	.20

Table 2: F1-score on the test set

The paper [1] introduced the LIAR dataset and also performed the same kind of study as the paper [4] on this dataset without using LIWC. They have used text properties and metadata (identity, Subject, political affiliation, place,...) provided by Liar to deliver a mix of the two ways mentioned above. The table 3 summarizes their models' performances.

In the last two years, a new kinds of studies were published in this subject. In 2019, the paper [5] published a study using news content, social context and spatiotemporal information for detecting fake news on social media. In this paper, they measured the social context of each news using user profiles, user posts, and network structures. In fact, user profiles on social media have been shown to be correlated with fake news detection [6]. Research has also shown that fake news pieces are likely to be created and spread by non-human accounts, such as social bots or cyborgs [7]. They also studies, in [5], the user posts on social media since people express their emotions or opinions towards fake news through social media posts, such as sensational reactions, etc. These features are important signals to study fake news and disinformation in general. Finally, they added

features from network users since users tend to form different networks on social media in terms of interests, topics, and relations, which serve as the fundamental paths for information diffusion. They used all of these features along with other textual features to obtain finally a high F1 score: **0.706** in the binary classification problem.

We can also mention the paper [8] where they extended the LIAR dataset by automatically extracting for each claim the justification that humans have provided to predict the labels. This extension improve efficiently the model to get a F1 score of **0.71** on the validation section of the binary problem and a F1 score of **0.38** on the problem with 6 classes. All of these latest studies introduced new features as the justification for each label or the meta-datas(identity, Subject, political affiliation, user profile...). In this project, we only focused on the content of the news and we used the elmo emebdding, which was not been used yet in this subject, with LIWC.

Models	valid	test
SVMs	.258	.255
Logistic regression	.257	.247
Bi-LSTMs	.223	.233
CNNs	.260	.270
Hybrid CNNs		
Text+Subject	.263	.235
Text+Speaker	**.277**	.248
Text+Job	.270	.258
Text+State	.246	.560
Text+Party	.259	.248
Text+Context	.251	.243
Text+History	.246	.241
Text+All	.247	**.274**

Table 3: Results obtained by [1]

3 Dataset

Politifact is a website created in 2007 and administered by journalists who check and label the veracity of promises and commitments made by American politicians. The particularity of Politifact is that each statement has a score ranging from 1 true to 6 pants-on-fire, thus making it possible to evaluate its veracity without necessarily falling into manicheism.
In 2017, the paper [1] introduced a new benchmark dataset LIAR with ten thousands of examples taken from POLITIFACT and labeled by humans for truthfulness, subject, context/venue, speaker, state, party, and prior history. This dataset adds metadata to the raw text, such as (identity, Subject, political affiliation, place, event...). We conducted our experiments on this dataset.
The table 4 gives a description of the dataset used. It is separated into three subsets, training, validation and test set containing respectively 10.240, 1284, 1267 statements for the training set, the validation set and the test set.

Dataset statistics	
Set	Set size
Training set	10240
Validation set	1284
Test set	1267
Avg. statement length	18.01
Top-3 Speaker	Size in the training set
Barack-Obama	488
Donald-Trump	273
Hillary-Clinton	239
Top-3 Speaker affiliation	Size in the training set
Republican	4497
Democrat	3336
None	1744

Class	Class size
True	1676
Mostly-true	1962
Half-true	2114
Barely-true	1654
False	1995
Pants-on-fire	839
Total	10240

Table 4: Presentation of the Liar dataset

4 LIWC

Linguistic Inquiry and Word Count (LIWC; Pennebaker, Booth, Francis, 2007) [2] is a word counting model that refers to a dictionary of grammatical, psychological, and content word categories. It is a paying api that we can use or download from their website [9]. The LIWC program analyses the input text with a group of built-in dictionaries. It compares each word in the text against a user-defined dictionary and identifies which words are associated with which dictionary. After, it calculates the percentage of total words that match each of the dictionary categories. It is all about a group of built-in dictionaries used to identify words and classify them. To clarify, we present an example extracted from their website. The word **cried** is part of five word categories: Sadness, Negative Emotion, Overall Affect, Verb, and Past Focus. Hence, if the word **cried** was found in the target text, each of these five subdictionary scale scores would be incremented. This tool used a basic idea, if a person is using a high rate of sad words, then he is sad. It gives for each sentence a vector of 93 features such as 'WC' (word count), 'pronoun', 'sad', 'social', 'power', etc. An example of a LIWC output is shown in figure 1. In this example, we give to the api the sentence 'I was sad when I had a fever'.

LIWC has been used to classify texts along psychological dimensions and to predict behavioural outcomes, making it a widely used text analysis tool in social sciences. LIWC can be seen as a processing tool applied to NLPs, specifically to text classification. The relative uses of the various categories of the LIWC may reflect the underlying psychology of demographics, honesty, health, status, relationship quality, group dynamics, or social context hidden in a text. LIWC analyses can provide information on a variety of psychological states and behaviours. It should be noticed that LIWC is really efficient if the text is between 50 and 500 words long. Combining LIWC categories with Machine Learning algorithms can yield interesting results on fake news detection.

text	WC	Analytic	Clout	Authentic	Tone	WPS	Sixltr	Dic	function	...	Comma	Colon	SemiC	QMark	Exclam	Dash	Quote	Apostro	Parenth	OtherP
I was sad when I had a fever	8	1,00	1,00	99,00	1,00	8,00	0,00	100,00	75,00	...	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Figure 1: Example of a LIWC output

5 Methods

5.1 General methodology

In this part, We will present the general principle of a text classification problem. The figure 2 presents this principle.

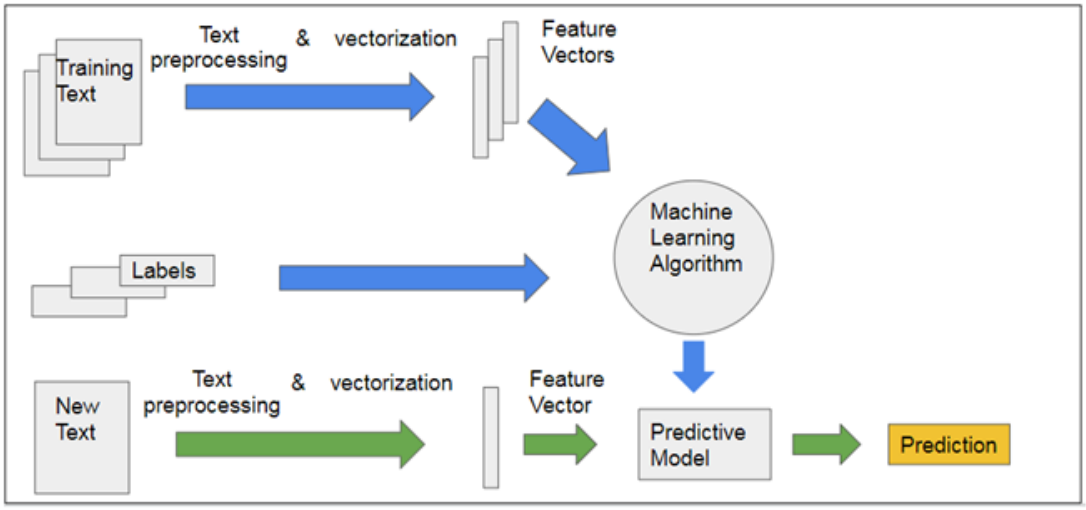

Figure 2: Principe générale de la classification de texte

In order to create a text classification model, the training data must first be prepared. Thus, the realization of such a model passes through 2 parts:

- First part is focused on the preparation of data. It is divided into two steps: first a pre-processing step of the corpus is done. This involves removing punctuation, unknown characters and so on, and then a second steps which consists of vectorization of the text, i.e.: moving from a text to a vector is performed.
- The second part is focused on the conception of a model based on statistical learning and trained on the vectorized data so as to have a predictive model.

5.2 Vectorization

In addition to LIWC presented in section 4, we have used other kinds of vectorization that we present in this section.

5.2.1 TF-IDF

Term Frequency-Inverse Document Frequency (tf-idf) is a weighting method that evaluates the importance of a token (a word or a punctuation mark) contained in a sentence relative to a corpus. The weight increases in proportion to the number of occurrences of the word in the document. It also change regarding the frequency of the word in the corpus. It promotes words that are used frequently in a document, but not so frequent outside of the document. It's calculated by the equations (1).

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}, \quad idf_i = log(\frac{|D|}{|\{d_k : t_i \in d_k\}|}) \quad et \quad tfidf_{i,j} = tf_{i,j} * idf_i \tag{1}$$

where $n_{i,j}$ is the number of occurrences of the token t_i in the sentence d_j and $|D|$ is the number of sentence in the corpus. this vectorization gives a sparse vectors with a lot of zeros.

5.2.2 Word2vec

Word2vec was created and published for the first time in 2013 by a team of researchers led by Tomas Mikolov at Google and patented [10]. It is a projection from a set of words into an embedding space vector. In this space words with similar meaning have similar vectors as the cosine similarity [1] between these two vectors is close to 1. In this project we have used different words embeddings.

- **Glove**: We have used glove embedding published by Jeffrey Pennington, Richard Socher, and Christopher D. Manning in 2014 [11]. In this paper they presented 4 embeddings: glove50d where each vector has 50 components, glove100d where each vector has 100 components, glove200d where each vector has 200 components, and glove300d where each vector has 300 components. In this project, we have tested all of them.
- **Elmo embeddings**: Elmo is a new words embedding method published by Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer in 2018 [12]. ELMo is a deep contextualized word representation that models both complex characteristics of word use (e.g., syntax and semantics), and how these uses vary across linguistic contexts (i.e., to model polysemy). These word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus.

5.3 Models used

In this section, we will present all the models used during this study.

5.3.1 Logistic regression

Logistic regression is a statistical model that uses in general the sigmoid function to model a binary dependent variable x and y. We suppose that the problem can be modeled

[1] Cosine similarity = <x,y>/$\|x\|\|y\|$

as :

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} = P(y = 1|x; \theta)$$

where θ is the parameter that we aim to optimize. For n pairs $(x_i, y_i)_{i=1,...,N}$ The goal of the logistic regression is to maximize the log-likehood with respect to θ :

$$\begin{aligned} l(\theta|X) &= \sum_{i=1}^{N} log(P(y_i|x_i; \theta)) = \sum_{i=1}^{N} log(h_\theta(x_i)^{y_i}(1 - h_\theta(x_i))^{1-y_i} \\ &= \sum_{i=1}^{N} y_i log(h_\theta(x_i)) + (1 - y_i) log(1 - h_\theta(x_i)) \end{aligned}$$

In general, we add a regularized term to avoid overffiting so the optimization problem can be written :

$$\min_\theta -\frac{1}{2N} \sum_{i=1}^{N} y_i log(h_\theta(x_i)) + (1 - y_i) log(1 - h_\theta(x_i)) + \frac{\lambda}{2N} \|\theta\|$$

where λ is the parameter of regularization. For each experience, we have chosen the parameter λ that gives the best f1 score on the validation set.

5.3.2 SVM

To use a svm classifier, The problem optimized is:

$$\min_{\theta,b} \frac{1}{N} \sum_{i=1}^{N} max(1, y_i(\theta^T x_i - b)) + \lambda \|\theta\|$$

SVM tries to finds the "best" margin (distance between the line and the support vectors) that split the classes regardless of the misclassification of some points, while logistic regression tries to not let any misclassification points. The advantage of svm is that it is more generalized for prediction than logistic regression model in some cases. SVM can then give better results in some cases but it takes a lot of time to run when we have high dimensions. In our project, we have used the **linear** kernel. For each experience, we have chosen the parameter λ that gives the best f1 score on the validation set. We can see in the figure 3 the difference between SVM and the logistic regression.

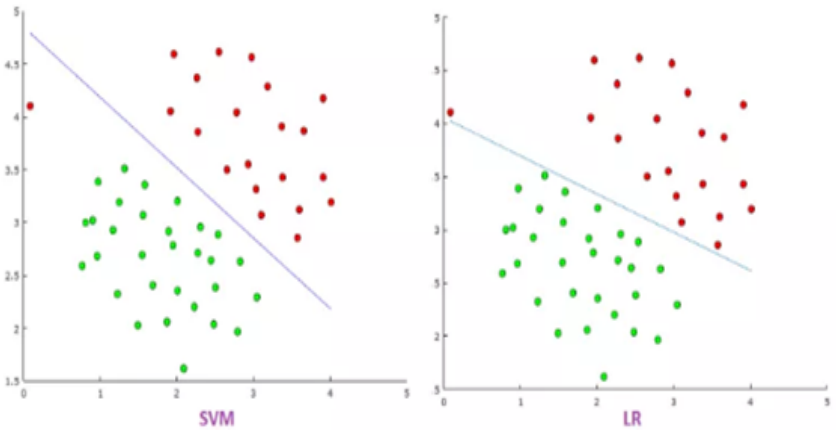

Figure 3: SVM vs Logistic Regression

5.3.3 Random Forest

Random Forest is a decision tree algorithm. It consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes the model's prediction. A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models. The figure 4 gives the principle of this model

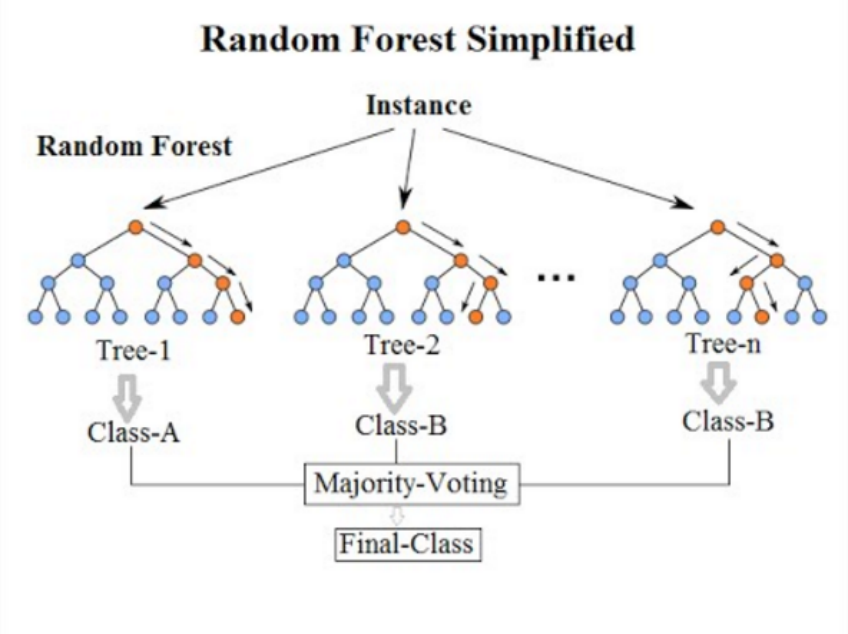

Figure 4: Random forest principle

The random forest model have a lot of hyperparameters. We presents here the most interesting of them:

- n_estimators : number of trees in the foreset
- max_features : max number of features considered for splitting a node
- max_depth : max number of levels in each decision tree
- min_samples_split : min number of data points placed in a node before the node is split
- min_samples_leaf : min number of data points allowed in a leaf node
- bootstrap : method for sampling data points (with or without replacement)

To fine-tune these hyperparameters, we have used the grid search method. For each parameter, we give a set of values and we tried all combination possible with these parameters by using cross validation. This method takes too much time to run, but it is useful to have good results. We didn't use this model with tf-idf, because we have a lot of columns in this case and random forest is not useful here. Furthermore, it takes too much time (so much more with grid search !). For each experience that uses the random forest classifier (with LIWC and word2vec) we fine-tuned these hyperparameters.

5.3.4 RNN

We have used the embedding of a sentence [2] as the input of the network. For the RNN, we have used the architecture presented in figure 5 with 'Relu' the activation function en each layer except the last we used the softmax function to get probabilities. We have used the optimizer Adam and the categorical_crossentropy loss function with 32 batchs with different epochs for each vectorization used.

Layer (type)	Output Shape	Param #
input_147 (InputLayer)	(None, 1117)	0
dense_275 (Dense)	(None, 512)	572416
dropout_235 (Dropout)	(None, 512)	0
dense_276 (Dense)	(None, 64)	32832
dense_277 (Dense)	(None, 6)	390

Total params: 605,638
Trainable params: 605,638
Non-trainable params: 0

Figure 5: Architecture of the Rnn used

5.3.5 CNN

Since convolutional neural networks architectures was introduced, CNNs have obtained the state-of-the art results on many text classification datasets. The filters sizes used for the CNN model was (2,3) with 64 filters in the two layers. Figure 6 shows the architecture of the convolutional neural network used.

Layer (type)	Output Shape	Param #
input_19 (InputLayer)	(None, 1117, 1)	0
conv1d_35 (Conv1D)	(None, 1117, 64)	192
average_pooling1d_11 (Averag	(None, 372, 64)	0
dropout_37 (Dropout)	(None, 372, 64)	0
conv1d_36 (Conv1D)	(None, 372, 64)	12352
average_pooling1d_12 (Averag	(None, 93, 64)	0
flatten_18 (Flatten)	(None, 5952)	0
dropout_38 (Dropout)	(None, 5952)	0
dense_38 (Dense)	(None, 256)	1523968
dense_39 (Dense)	(None, 1)	257

Total params: 1,536,769
Trainable params: 1,536,769
Non-trainable params: 0

Figure 6: Architecture of the Cnn used

[2] The mean of each word embedding in the sentence

6 Experiments

6.1 Features selection applied to LIWC

LIWC gives 93 features. So we tested in this case 2 methods of features selection using the regression logistic classifier. In the first method, we select features with respect to the importance of each coefficient in the logistic regression formula and the second by plotting the accuracy score with respect to the features used. In the second method, we started with the first feature, then, we add the second feature and so on until all the features are used. After that, we delete each feature where the accuracy score decrease with respect to the last score.

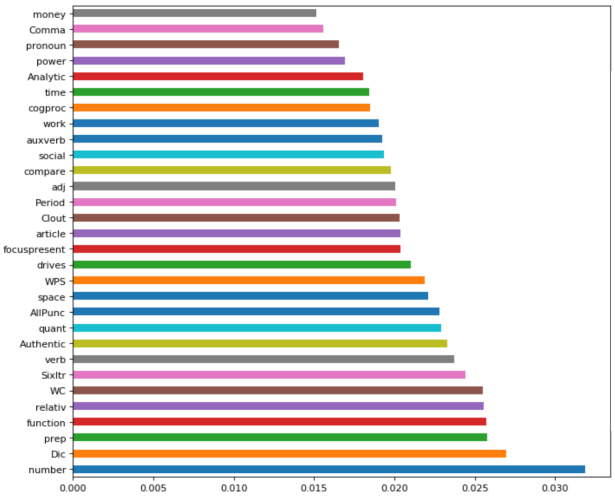

Figure 7: Features selection with the first method

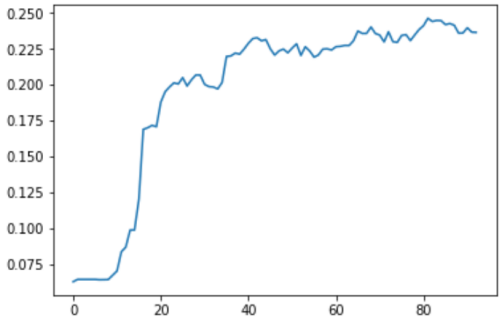

Figure 8: Features selection with the second method

The figure 7 presents the histogram of the importance of each feature selected. The feature with the longest bar is the most important and so on. the x-axis presents the values taken by coefficients in the regression logistic formula. The figure 8 presents the evolution of the accuracy score with respect to the number of features used. The x-axis presents the number of features used and the y-axis presents the accuracy score. We have used the features in the same order given by LIWC. We can see that sometimes adding a feature can decrease the score. It is these features that we have deleted. We obtained the following f1 scores on the validation set :

- All features : .228
- Features selection with the first method : .163
- Features selection with the second method : **.236**

We can see that the second method can improve the f1 score.

6.2 Results on the entire dataset with 6 classes

We used in this project two types of approaches, a global one with all 6 classes and a binary one. For the binary classification, we set the label on "true" for all news labeled

"true", "mostly-true" and "half-true" and label "false" for fake news labeled "barely-true", "false", "pants-on-fire". We have used the same models for the two approaches. We have compared the results obtained in [4] and [1] with our own ones using the F1-score on the validation set. Note that [4] uses the macro F1-score. Given the unbalanced distribution of labels in the dataset, we find it more useful to use the weighted F1-score.

	Tf-idf	LIWC	tf-idf+LIWC	glove 50d	100d	200d	300d	Elmo	Elmo+LIWC
Reg-log	.245	.236	.256	.218	.223	.236	.232	.266	**.270**
SVM	.240	.219	.259	.192	.212	.216	.218	.248	**.264**
RF		.211		.215	.205	.217	.205	.234	**.238**
RNN		.250		.247	.252	.257	.261	.273	**.281**
CNN		.229		.218	.230	.248	.258	**.281**	.254

Table 5: F1 score on the validation set with 6 classes

	Tf-idf	LIWC	tf-idf+LIWC	glove 50d	100d	200d	300d	Elmo	Elmo+LIWC
Reg-log	.223	.238	.255	.242	.247	.237	.234	.260	**.262**
SVM	.242	.222	.248	.194	.237	.220	.232	.244	**.256**
RF		.196		.217	.198	.192	.197	.202	**.209**
RNN		.229		.262	.242	.252	.248	.256	**.274**
CNN		.241		.231	.234	.248	.221	.249	**.254**

Table 6: F1 score on the test set with 6 classes

From the two tables 5 and 6, we can see that the use of embeddings is more efficient and gives more interesting results. We see also that adding LIWC features ameliorate the models specially when we use Machine Learning algorithm's (Reg-log, SVM, RF) .
The use of Elmo embeddings allows to obtain best scores. We can benefit even more from using these embeddings by taking into account the embeddings of each word seperatly and not the average of the embeddings. To achieve this we need a powerful gpu which is not the case for this project.

Our best model are in the same range as results in [1]. The table 7 resumes the comparison between our best model and the best results from [4, 1]. Our best model is not compared to results in [8]: **.38** on the validation set. In fact, this paper uses the justification that humans have provided to predict the true label and in our work we only focused on the text from the news. We can see that our model outperforms the other model that use the same kind of features without adding the justification of labels as in [8].

model	F1 score	
	val	test
our best model	**.281**	**.274**
model from [1]	.277	.248
model from [4]	.220	.200

Table 7: Comparison of our results with other studies on the 6 classes classification problem

6.3 Results on the entire dataset with 2 classes

This problem is easier and so we obtain better results. We have done the same experiences presented before and obtained the results presented in tables 8 and 9:

	Tf-idf	LIWC	tf-idf+LIWC	glove 50d	100d	200d	300d	Elmo	Elmo+LIWC
Reg-log	.621	.586	.618	.591	.598	.613	.612	**.637**	.618
SVM	.611	.544	.600	.568	.590	**.621**	.608	.603	.617
RF		.586		.610	.610	.605	**.611**	.605	.601
RNN		.609		.630	.628	**.645**	.635	.631	.627
CNN		.600		.616	.627	.625	.624	**.632**	.630

Table 8: F1 score on the validation set with 2 classes

	Tf-idf	LIWC	tf-idf+LIWC	glove 50d	100d	200d	300d	Elmo	Elmo+LIWC
Reg-log	.608	.584	.610	.591	.593	.598	.600	.607	**.613**
SVM	.603	.561	.619	.582	.581	.600	.601	**.620**	**.620**
RF		.604		.584	.591	.601	**.606**	.590	.587
RNN		.600		.610	.610	.603	.614	**.622**	.618
CNN		.593		.603	.608	.597	.600	.591	**.614**

Table 9: F1 score on the test set with 2 classes

Unlike the results in the 6 classes classification problem, the use of Elmo+LIWC does not gives the best results on the validation set in the binary classification problem. We have obtained the best result on the validation set with a RNN using glove200d word2vec embeddings and the best test score with a RNN using Elmo embeddings. But in the test set, adding LIWC ameliorate the results in same cases and gives almost the same scores as the best score. In this case, LIWC is not as efficient as in the 6 classes classification problem.

Our best model, in this case, outperforms the models presented in [4]: **.645** vs 0.58 on the validation set. The paper [1] does not do the studies with the binary classification. As for the 6 classification problem, our model in the binary classification is outperformed by the model in [5]: **.706** on the validation set the model in [8]; **0.71** on the validation set. We recall that [5] add to textual features social context features and [8] add to textual features the justification that humans have provided to predict the true labels.

To go further with the binary classification, we have sub-divided the datasets with respect to different criteria and study the fake news detection on the sub-datasets obtained.

6.4 Results on datasets' sub-divisions

In addition to the comparative work carried out previously, we conducted a study using tf-idf and LIWC on sub-divisions of the data set grouped by political affiliation of the speakers and by speakers in order to introduce a psychological aspect. In fact, LIWC introduce a psychological aspect to words and it has been conceived to analyse texts written by the same person. In all experiences presented before and researchers presented in the related work, LIWC was used on datasets with different speakers. This has an impact on the interpretability of the results. Thus, to have better interpretability we have used the subsets presented before. We have focused in this studies on the binary classification problem and used only regression logistic.

6.4.1 Sub-divisions grouped by speaker

For this studies, we have taken only speakers that have more than 100 statements in the training set. Thus, we can apply a machine learning model even though 100 is not sufficient in general. The table 10 presents the authors preserved and the number of their statements in the data sets.

Authors	#training set	#validation set	#test set
Barack-Obama	488	61	62
Donald-Trump	273	37	33
Hillary-Clinton	239	27	31
Mitt-Romney	176	19	17
Scott-Walker	149	17	17
John-Mccain	148	20	21
Chain-Email	142	23	13
Rick-Perry	142	14	17
Marco-Rubio	117	22	14
Rick-Scott	115	14	21

Table 10: Sub-datasets' by speaker

We have used only regression logistic since it is more efficient with very small datasets. We used, in addition to the F1 score, the accuracy score on the validation and the test set to compare and analyse the results. The table 11 presents the results obtained. In this table, we can see that in the validation set, the logistic regression with LIWC outperforms the logistic regression with tf-idf on all subsets except for the "Donald-Trump" sub-dataset in term of F1 measure. It is also the case in term of accuracy score in the validation set except for "Barack-Obama" and "Donald-Trump" datasets'. These results prove that , in this case, the use of LIWC is very useful than tf-idf. We can also see that LIWC on texts from the same speaker is more useful and the interpretability of the features is better.

	F1-score				accuracy_score			
Authors	Tf-idf		LIWC		Tf-idf		LIWC	
	val	test	val	test	val	test	val	test
Barack-Obama	.626	.697	**.680**	.710	**.737**	.790	.704	.741
Donald-Trump	**.602**	.385	.582	.507	**.675**	.545	.648	.606
Hillary-Clinton	.708	.589	**.796**	.668	.777	.709	**.814**	.709
Mitt-Romney	.437	.671	**.678**	.519	.526	.705	**.684**	.529
Scott-Walker	.535	.376	**.596**	.647	.529	.470	**.588**	.647
John-Mccain	.500	.895	**.853**	.558	.500	.904	**.850**	.523
Chain-Email	**1.00**	.775	**1.00**	.775	**1.00**	.846	**1.00**	.846
Rick-Perry	.766	.698	**.786**	.529	**.785**	.705	**.785**	.529
Marco-Rubio	.458	.691	**.630**	.735	.545	.785	**.636**	.714
Rick-Scott	.514	.489	**.571**	.617	**.571**	.523	**.571**	.619

Table 11: Results on sub-datasets' by speaker

6.4.2 Sub-divisions grouped by political affiliation

In the liar dataset, we have several political parties. In this study, we conserve only the Democratic Party, the Republican Party and 'None' since they are the most represented in the dataset ('None' is the label for speakers without party). We have separated each set of our dataset into three separate subsets. The table 12 presents these parties and the number of their statements in the data sets.

Party affiliation	#training set	#validation set	#test set
Republican	4497	597	571
Democrat	3336	395	406
None	1744	233	214

Table 12: Sub-datasets' by political affiliation

	F1-score				accuracy_score			
Party affiliation	Tf-idf		LIWC		Tf-idf		LIWC	
	val	test	val	test	val	test	val	test
Republican	**.641**	.634	.610	.610	**.643**	.637	.620	.610
Democrat	**.612**	.643	.606	.600	**.665**	.694	.600	.590
None	**.669**	.562	.421	.492	**.668**	.565	.511	.574

Table 13: Results on sub-datasets' by party affiliation

The table 13 summarize the performances obtained. We can conclude from this table that separating the data set according to the political affiliations of the speakers doesn't improve the performance of the model unlike the separation by the speaker that improves the model. So, we can see from these two experiences that LIWC can help in fake news detection if the studies is focused on a specific speaker. It is logic since LIWC help to introduce a psychological aspect of words, so it is more adapted to separate the data by speaker than party affiliation.

7 Conclusion

In this project, we have conducted two types of studies. The first study concern the classification problem with 6 classes and 2 classes. We have demonstrated that the use of LIWC is useful in the 6 classes problem and gives the best model in term of f1 score in the validation set when it is concatenated to the Elmo embeddings. It is not the same results on the 2 classes problem. In this case, adding LIWC doesn't improve the model and we get the best model using glove200d word2vec embeddings. In this study, we outperformed with our best model in each case the models presented in the related works that used the same kind of features (only textual features).

The second study of our project demonstrated the utility of using LIWC on news from the same speaker. We have shown that for datasets with the same speaker, LIWC gives a high f1 score in the validation set than tf-idf. That wasn't the case for datasets grouped by affiliation since it is from different speakers. In this part, we have also seen that the classification highly dependent of the speaker. We can conclude this also by the studying in [5] since adding a social context features by studying the speakers gives much better results.

We have shown in this study that it is complicated to get high scores in the fake news detection problem as the classic sentiment analysis problem. It is a challenging task, but the use of features that introduce a psychological, a context and a rhetorical aspects can give very interesting results.

References

[1] William Yang Wang. "liar, liar pants on fire": A new benchmark dataset for fake news detection. 2017.

[2] Yla R. Tausczik and James W. Pennebaker. The psychological meaning of words: Liwc and computerized text analysis methods. 2010. URL `http://homepage.psy.utexas.edu/homepage/students/Tausczik/Yla/index.html`.

[3] Victoria L. Rubin Niall J. Conroy and Yimin Chen. Automatic deception detection: Methods for finding fake news. 2016.

[4] Jin Yea Jang Svitlana Volkova Yejin Choi Hannah Rashkin, Eunsol Choi. Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking. 2017.

[5] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media. *CoRR*, abs/1809.01286, 2018. URL `http://arxiv.org/abs/1809.01286`.

[6] Kai Shu, Suhang Wang, and Huan Liu. Exploiting tri-relationship for fake news detection. *CoRR*, abs/1712.07709, 2017. URL `http://arxiv.org/abs/1712.07709`.

[7] Kai Shu, Suhang Wang, Thai Le, Dongwon Lee, and Huan Liu. Deep headline generation for clickbait detection. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pages 467–476. IEEE Computer Society, 2018. doi: 10.1109/ICDM.2018.00062. URL `https://doi.org/10.1109/ICDM.2018.00062`.

[8] Tariq Alhindi, Savvas Petridis, and Smaranda Muresan. Where is your evidence: Improving fact-checking by justification modeling. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 85–90, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5513. URL `https://www.aclweb.org/anthology/W18-5513`.

[9] Liwc website. URL `https://liwc.wpengine.com/`.

[10] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method, 2014. URL `http://arxiv.org/abs/1402.3722`. cite arxiv:1402.3722.

[11] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL `https://www.aclweb.org/anthology/D14-1162`.

[12] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018. URL `http://arxiv.org/abs/1802.05365`.