# Master thesis

Simon Lorent

August 7, 2019

# Contents

# Master thesis

Fake news detection using machine learning

**Simon Lorent**

## Acknowledgement

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

# Master thesis

Fake news detection using machine learning

**Simon Lorent**

## Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

# Chapter 1

# Introduction

## 1.1 What are fake news?

### 1.1.1 Definition

Fake news have quickly become a society problem, being used to propagate false or rumorus informations in order to change behaviors of peoples. Before stating to work on detecting fake news, it is needed to first understand what they are. It have been show that propagation of fake news have had a non negligable influence of 2016 US presidential elections[1]. A few facts on fake news in the United States:

- 62% of US citizen get there news for social medias[2]
- Fake news had more share on facebook than mainstream news[3].

Fake news have also been used in order to influence the referendum in the United Kingdom for the "Brexit".

In this paper I experiment the possibility to detect fake news based only on textual information by applying traditional machine learning techniques[4, 5, 6] as well as bidirectional-LSTM[7] and attention mechanism[8] on two different dataset that contains different kinds of news.
There are two aspects of fake news detection that need to be taken into account according to Shu et al[9]. The first is characterization or what are fake news and the second is detection. In order to build detection models, it is need to start by charaterization, indeed, it is need to understand what are fake news before trying to detect them.

Fake news definition is made of two part: authenticity and intent. Authenticity means that fake news content fale information that can be verified as such, which means that concpiracy theory is not included in fake news as there are difficult to be proven true or false in most cases. The second part, intent, means that the false information have been written with the goal of missleading the reader.

### 1.1.2 Fake News Characterization

**Definition 1** *Fake news is a news article that is intentionally and verifiably false*

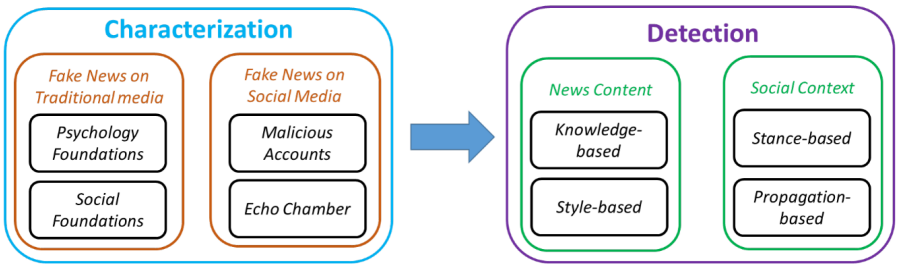



Figure 1.1: Fake news on social media: from characterization to detection.[9]

The part of the definition introducing the intent of missleading the reader automaticaly discard satire news medias, that is why this works will focus on the first part, the fact that the piece of information is verifiably false or true. Indded, even if satire news medias does not have the intent to misslead the readers, not all of them have the ability of making criticisme and not taking it to the first degree. On the other hand, in the case of political media, even if it clearly try to influence the consumer, verifying the authenticity of there claims is usualy harder as, in most of the cases, openly lies.

## 1.2 Feature Extraction

### 1.2.1 News Content Features

Now that fake news have been defined and the target have been set, it is needed to analize what features can be used in order to classify fake news. Starting by looking at news content, it can be seen that it is made of four principal raw components:

- **Source**: Where does the news come from, who wrote it, is this source reliable or not.
- **Headline**: Short summary of the news content that try to attract the reader.
- **Body Text**: The actual text content of the news.
- **Image/Video**: Usualy, textual information is agremented with visual information such as images, videos or audio.

Features will be extracted from these four basic components, with the mains features being linguistic-based and visual-based. As explained before, fake news are used to influence the consumer, and in order to do that, they often use a specific language in order to attract the readers. On the other hand, non fake news will mostly stick to a different language register, being more formal. This are linguistic-based features, to which can be added lexical features such as total number of words, frequency of large words or unique words.

The second features that need to be taken into account are visual features. Indeed, modified images are often used to add more weight to the textual information. For example, the **Figure 1.2** is supposed to show the progress of deforestation, but the two images are acutaly from the same original one, and in addition the WWF logo make it looks like to be from a trusted source.


Figure 1.2: The two images provided to show deforestation between two date are from the same image taken at the same time.

### 1.2.2 Social Context Features

In the context of news sharing on social medias, multiples aspect can be taken into account, such as user aspect, post aspect and group aspect. For instance, it is possible to analyize the behavior of specific users and use their metadata in order to find if a user is at risk of trusting or sharing false information. For instance, those metadata can be its center of intereset, its number of followers, or anything that relates to it.

Post-based aspect is in a sense similar to user based: it can use post metadata in order to provide usefull informations, but in addition to metadata, the actual content can be used. It is also possible to extract features from the content using latent Dirichlet allocation (LDA)[10].

## 1.3 News Content Models

### 1.3.1 Knowledge-based models

Now that the differents kinds of features available for the news have been defined, it is possible to start to explain what kind of models can be built using these features. The first models that relates to the news content is based on knowledge: the goal of this model is to check the truthfulness of the news content and can be acheived in tree differents way (or a mixture of them):

- **Expert-oriented**: relies on expert, such as journalist or scientist, to asses the news content.
- **Crowdsourcing-oriented**: relies on the wisdom of crowd that says that if a suficiently large amount of persons says that something is false or true then it should be.
- **Computational-oriented**: relies on automatic fact checking, that could be based on external resources shuch as DBpedia.

These methods all have pros and cons, hiring expert might be constly, and expert are limited in number and might not be able to treat all the news that are produced. In the case of crowdsourcing it can easily be fooled if enought bad annotators breaks the system and automatic fact checking might not have the necessary accuracy.

### 1.3.2 Style-based model

As explained earlier, fake news usualy tries to influence consumier behavior, and thus generaly use a specific style in order to play on the emotion. These methods are called deception-oriented stylometric methods.

The second method is called objectivity-oriented approaches and tries to capture the objectivity of the texts or headlines. These kind of style is mostly used by partisan article or yellow-journalism, that is, website that relies on eye-catching headline without reporting any useful information. An example of these kind of headline could be:

> You will never beleive what he did !!!!!!

This kind of headline plays on the curiosity of the reader that would click to read the news.

## 1.4 Social Context Models

The last features that have not been used yet are social media features. There are two approaches to use these features: stance-based and propagation-based.

**Stence-based** approaches use implicit or explicit representation. For instance, explicit representation might be positive or negative votes on social medias. Implicit representation need to be extracted from the post itself.

**Propagation-based** approaches use features related to sharing such as the number of retweet on twitter.

## 1.5 Related Works

### 1.5.1 Fake news detection

Current research focus mostly on using socials features and speaker informations in order to improve the quality of classifications.

Ruchansky et al.[11] proposed an hybrid deep model for fake news detection making use of multiple kind a features such as temporal engagement between n users and m news-articles over time and produce a label for fake news categorisation but as well a score for suspicious users.

Long et al. [12] proposed to apply attention mechanism to the speaker profile in an hybrid way with an LSTM on the textual information.

Tacchini et al.[13] proposed a method based on social network informations such as likes and users in order to find hoax information.

Thorne et al.[14] proposed a stacked ensemble classifier in order to adress a subproblem of fake news dectection which is stance classification. It is the fact of finding if an articile agree, disagree or simply discus a fact.

Granik and Mesyura[15] used Nave-Bayes classifier in order to classify news from buzzfeed dataset.

In addition to text and social features, Yang et al.[16] used visual features such as images with a convolutional neural network.

Wang et al.[17] also used visual features for classifying fake news but uses adversarial neural network to do so.

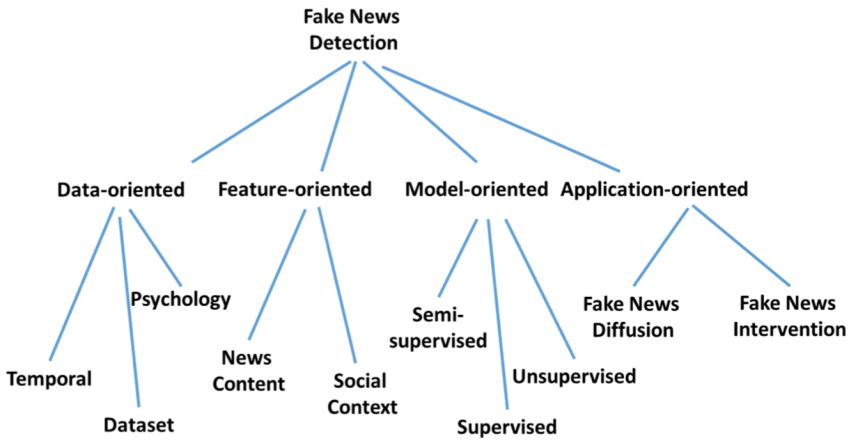



Figure 1.3: Differents approaches to fake news detection.

### 1.5.2 State of the Art Text classification

There are two mains categories of state of the art that are interesting for this work: previous work on fake news detection and on general text classification. Works on fake news detection is almost inexistant and mainly focus on 2016 US presidential elections or does not uses the sames features. That is, when this work focus on automatic features extraction using machine learning and deep learning, other works make use of hand crafted features[18, 19] such as psycholinguistic features[20] which is not the goal here.

When it comes to state of the art for text classifiction, it includes Long short-term memory (LSTM)[7], Attention Mechanism[21], IndRNN[22], Attention-Based Bidirection LSTM[8], Hierarchical Attention Networks for Text Classification[23], Adversarial Training Methods For Supervised Text Classification[24], Convolutional Neural Networks for Sentence Classification[25] and RMDL: Random Multimodel Deep Learning for Classification[26]. All of these models have comparable performances.

## 1.6 Conclusion

As it have been shown in **Section 1.2** and **Section 1.3** mutliples approaches can be used in order to extract features and use them in models. This works focus on textual news content features. Indeed, other features related to social medias are difficult to aquire. For example, users information are difficult to obtain on Facebook, as well as post information. In addition, the different datasets that have been presented at **Section 2.2** does not provides any other information than textual ones.

Looking at **Figure 1.3** it can be seen that the main focus will be made on unsupervised and supervised learning models using textual news content. It should be noted that machine learning models usualy comes with a trade-off between precision and recall and thus that a model which is very good at detected fake news might have a high false positive rate as opposite to a model with a low false positive rate which might not be good at detecting them. This cause ethical questions such as automatic censorship that will not be discused here.

# Chapter 2

# Data Exploration

## 2.1 Introduction

A good starting point for the analysis is to make some data exploration of the data set. The first thing to be done is statistical analysis such as counting the number of text per class or counting the number of words per sentence. The second step consit of doing Latent Dirichlet Allocation[10] in order to make unsupervised clustering of the text and see if there is some kind of correlation between the clusters to which a text belongs and its labels.

## 2.2 Datasets

### 2.2.1 Fake News Corpus

This works uses multiples corpus in order to train and test different models. The main corpus used for training is called Fake News Corpus[27]. This corpus have been automatically crawled using `opensources.co` labels. In other words, domains have been labeled with one or more labels in

- Fake News
- Satire
- Extreme Bias
- Conspiracy Theory
- Junk Science
- Hate News
- Clickbait
- Proceed With Caution
- Political
- Credible

These annotations have been provided by crowdsourcing, which means that they might not be exactly accurate, but are expected to be close to the reality. Because this works focus on fake news detection against reliable news, only the news labels as fake and credible have been used.

### 2.2.2 Liar, Liar Pants on Fire

The third and last dataset is **Liar, Liar Pants on Fire** dataset[28], which is a collection of twelve thousand small sentences collected from various sources and hand labeled. They are devided in six classes:

- pants-fire
- false
- barely-true
- half-true
- mostly-true
- true

This set will be used a second test set. Because in this case there are six classes againt two in the other cases, a threshold should be used in order to fix which one will be concidered as true or false in order to be compared with the other dataset.

It should be noted that this one differ from the two other dataset is it is composed only on short sentences, and thus it should not be expected to have very good results on this dataset for models trained on Fake News Corpus which is made of full texts. In addition, the texts from the latest dataset are more politicaly oriented then the ones from the first one.

## 2.3 Dataset statistics

### 2.3.1 Fake News Corpus

#### General Analysis

Because **Fake News Corpus** is the main dataset, the data exploration will start with this dataset. And the first thing is to count the number of items per class. Before starting the analysis, it is needed to cleanup the dataset. As it is originaly given in a large 30GB CSV file, the first step is to put everything in a database in order to be able to retreive only wanted peice of information. In order to do so, the file have been red line by line. It appears that some of the lines are baddly formated, preventing them to be red correctly, in this case they are dropped without being put in the database. Also, each line that is a duplicate of a line already red is also dropped. The second step in cleaning the set consist of some more duplicate removal. Indeed, dropping same lines remove only exact duplicate. It appears that some news does have the same content, with slight variation in the title, or a different author. In order to remove the duplicate, each text is hashed using SHA256 and those hash a compared, removing duplicates and keeping only one.

Because the dataset have been cleaned, numbers provided by the dataset creators and number computed after cleaning will be provided. We found the values given at **Table 2.1**. It shows that the number of fake news is smaller by a small factors with respect to the number of reliable news, but given the total number of items it should not cause any problems. But it will still be taken into account later on.

| Type | Provided | Computed |
|---|---|---|
| Fake News | 928,083 | 770,287 |
| Satire | 146,080 | 855,23 |
| Extreme Bias | 1,300,444 | 771,407 |
| Conspiracy Theory | 905,981 | 495,402 |
| Junk Science | 144,939 | 79,342 |
| Hate News | 117,374 | 65,264 |
| Clickbait | 292,201 | 176,403 |
| Proceed With Caution | 319,830 | 104,657 |
| Political | 2,435,471 | 972,283 |
| Credible | 1,920,139 | 1,811,644 |

Table 2.1: Number of texts per categories

In addition to the numbers provided at **Table 2.1**, there are also two more categories that are in the dataset but for which no description is provided:

- Unknwon: 231301
- Rumor: 376815

To have a better view of the distribution of categories, an histogram is provided at **Figure 2.1**.
In addition, the number of words per text and the average number of words per sentences have been computed for each text categories. **Figure 2.2** shows the boxplots for these values. It can be seen that there is no significative difference that might be used in order to make class prediction.
Before counting the number of words and sentences, the texts are preprocessed using gensim[29] and NLTK[30]. The first step consist of spliting text into an array of sentences on stop punctuation such as dots or questions mark, but not on commas. The second step consist on filttering words that are contained in these sentences, to do so, stopwords (words such as 'a', 'an', 'the'), punctuation, words or size less or equal to tree, non alpha-numeric words, numeric values and tags (such as html tags) are removed. Finnaly, the number of word still present is used.

An interesting features to look at is the distrubtion of news sources with respect to their categories. It shows that in some case some source a predominant. For instance, looking at **Figure 2.3** shows that most of the reliable news are from *nytimes.com* and in the same way, most of the fake news are coming from *beforeitsnews.com*. That have to be taken into account when training and testing models as the goal is not to distinguish between these two sources but between fake news and reliable news.

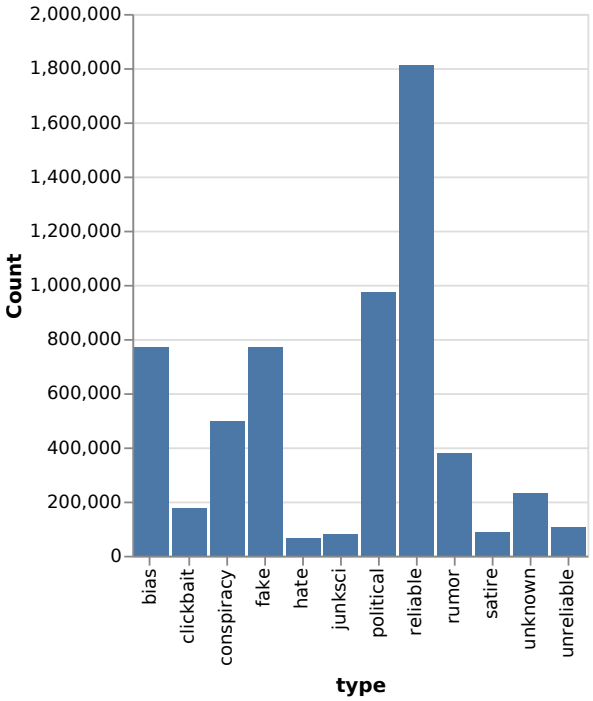
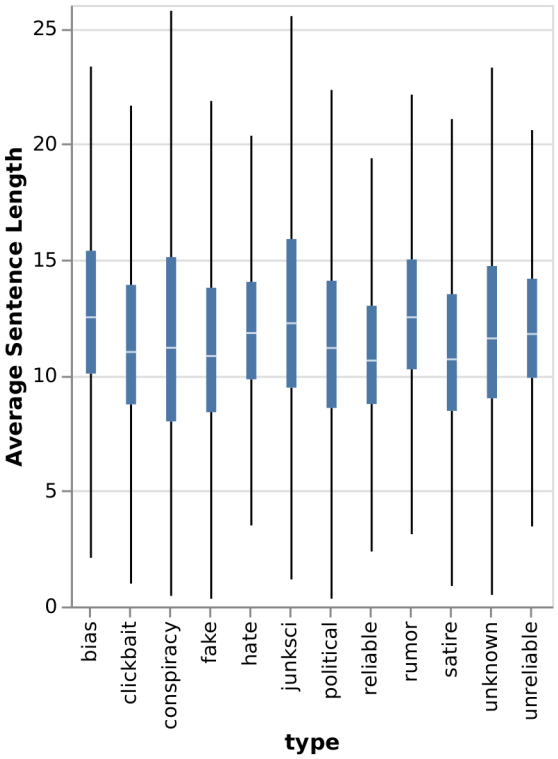
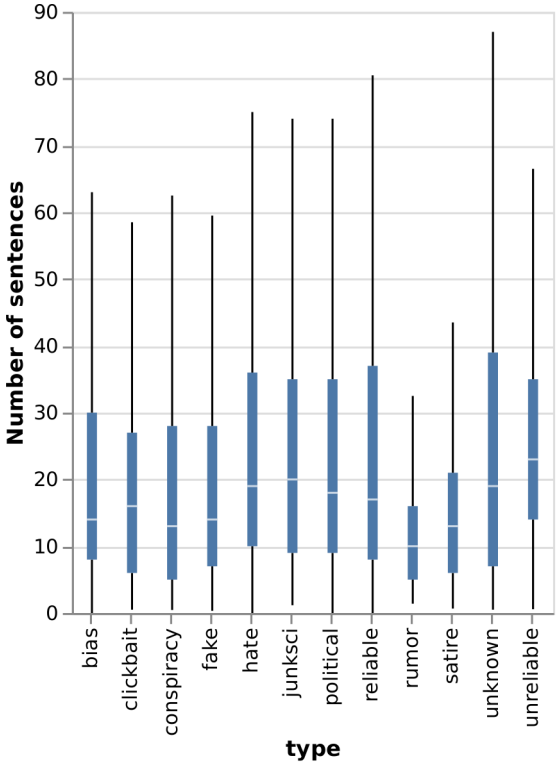



Figure 2.1: Histogram of text distribution along their categories on the computed numbers.

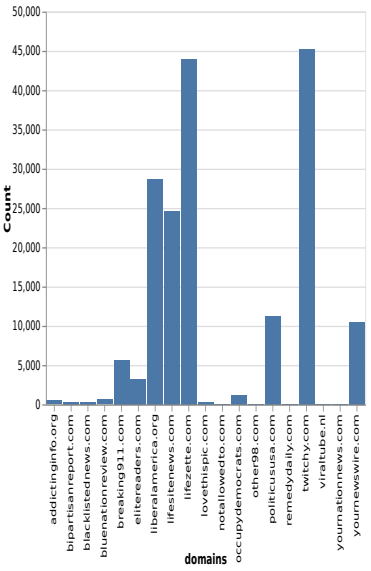



(a) Boxplot of average sentence length for each category.

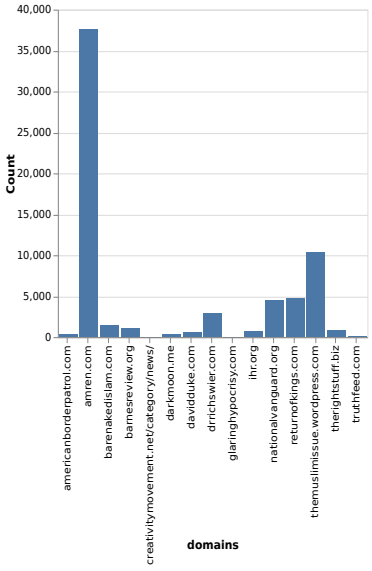



(b) Boxplot of number of sentences for each category.

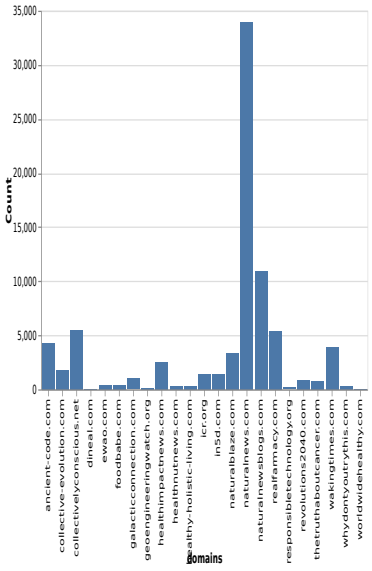
Figure 2.2: Summary statistics



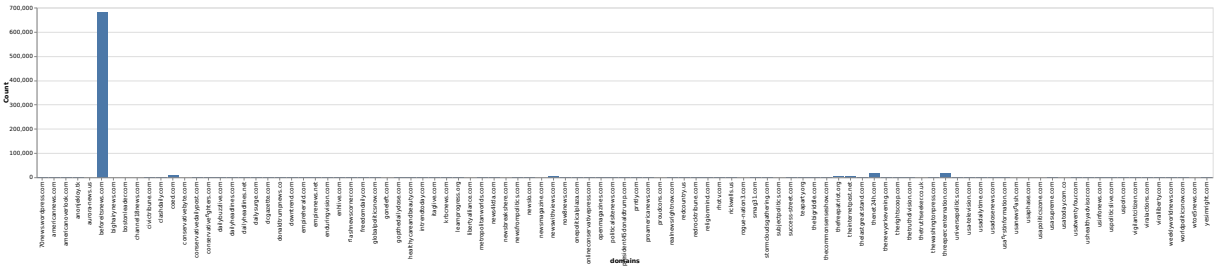

(a) Clickbaits



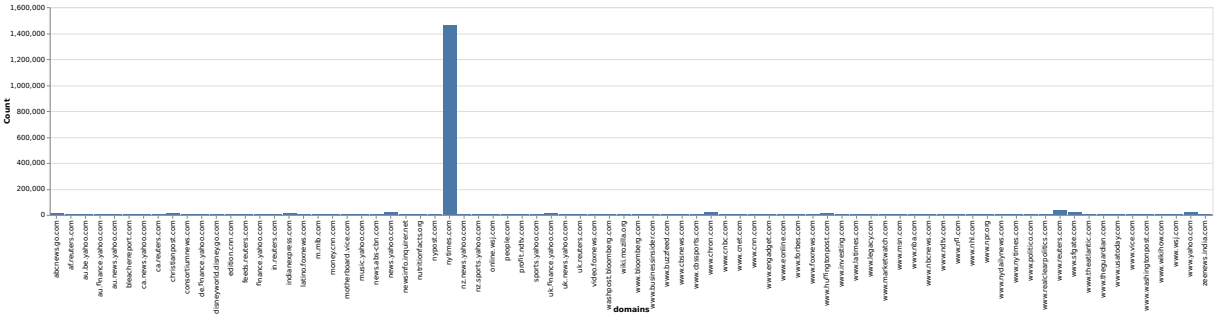

(b) Hate




(c) Junk sience




(d) Fake

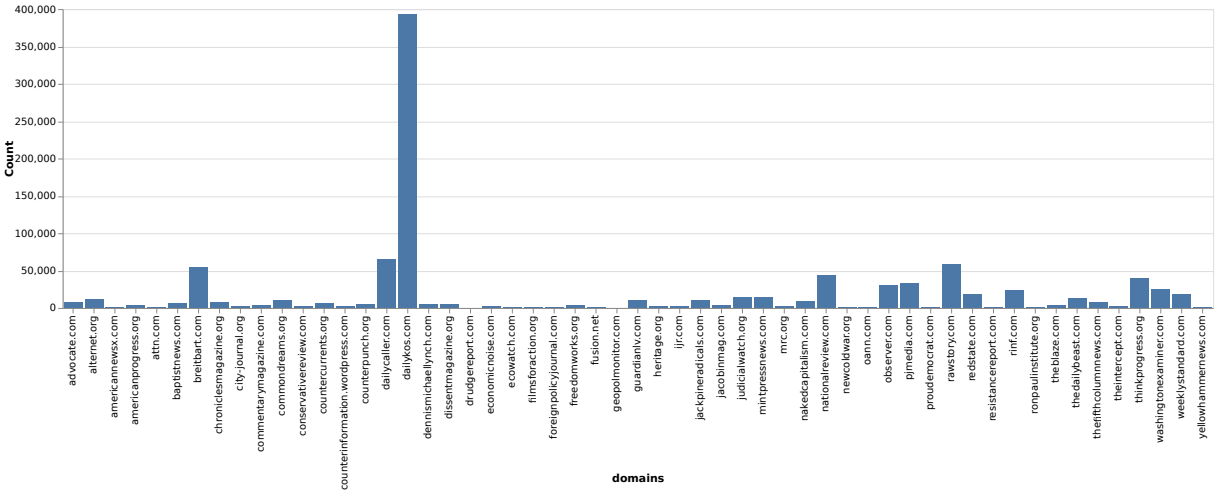



(e) Reliable

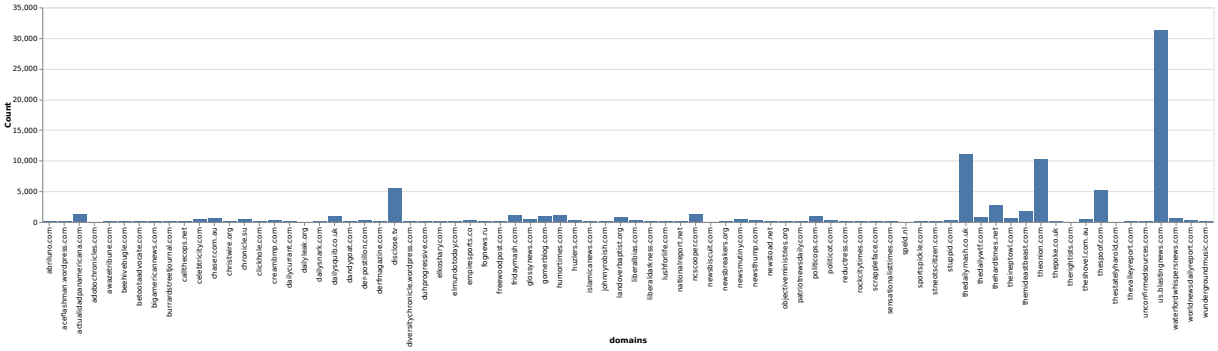
An other import feature to look at is the distribution of the number of words in the text. Indded, at some point it will be needed to fix a constant length for the texts and using to small length would mean a lot of cutting and using too long size would mean too much padding. It is thus needed to investigate the length of the texts in order to choose the right one. It can be seen at **Figure 2.4** that reliable news have slightly more words than fake news, but the difference is minimal.

### Fake News analysis

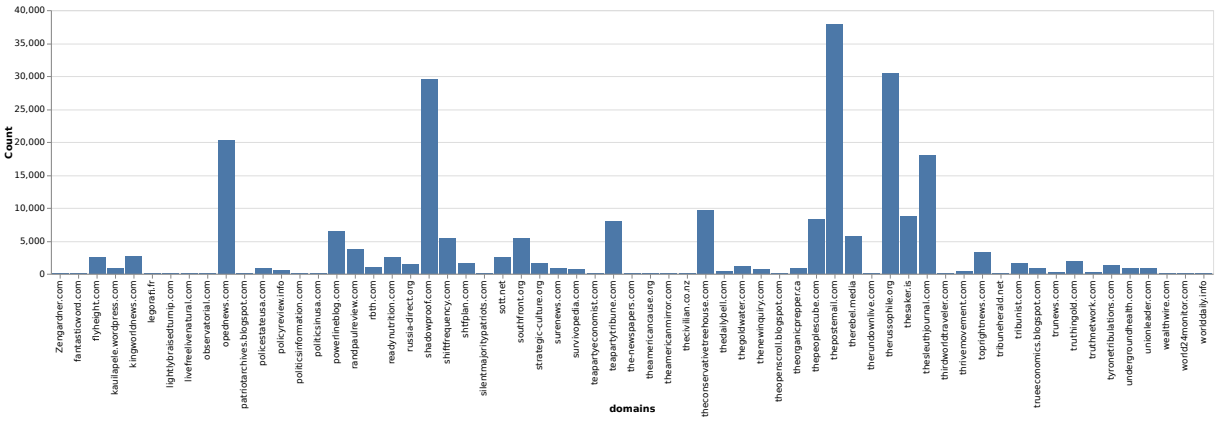
In this section, the analysis will focus on fake news and reliable news. Because of what shows **Figure 2.3**, that is, some categories are almost all from the same source, an analysis of what happens when dropping these sources. First, lets compare the number of news while and while not taking into account majors sources.
Comparing **Figure 2.5** and **Figure 2.6** shows that even by removing *nytimes.com* and




(f) Political




(g) Satire




(h) Unknown

Figure 2.3: Histogram of news origin for each categories.

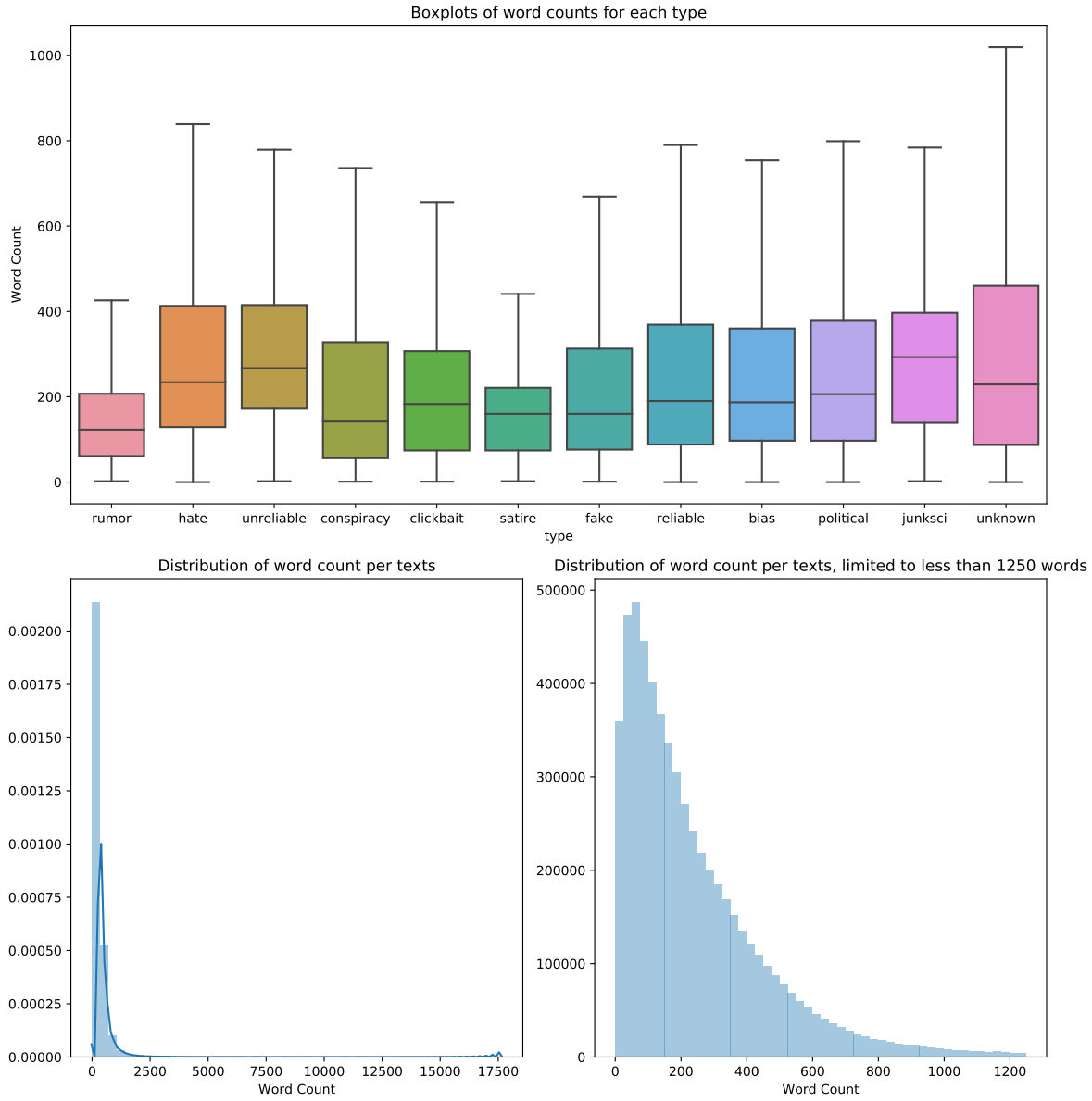



Figure 2.4: Distribution of the number of words per text

*beforeitsnews.com* news from the dataset still leave enought texts to train the different models without the risk of learning something unwanted such as only separating these two sources. But one drawback is that the ratio between fake news and reliable news is going from around one half to around one fourth.

### 2.3.2 Liar-Liar corpus

As said in **Chapter 1**, this dataset is made of small text of one or two sentences at most. Which means that they are smaller than than the ones in *fake news corpus*. The distribution of words length can be seen at **Figure 2.7**. In addition, this dataset is not unbalanced as the other corpus is, which means that precautions does not have to be taken.

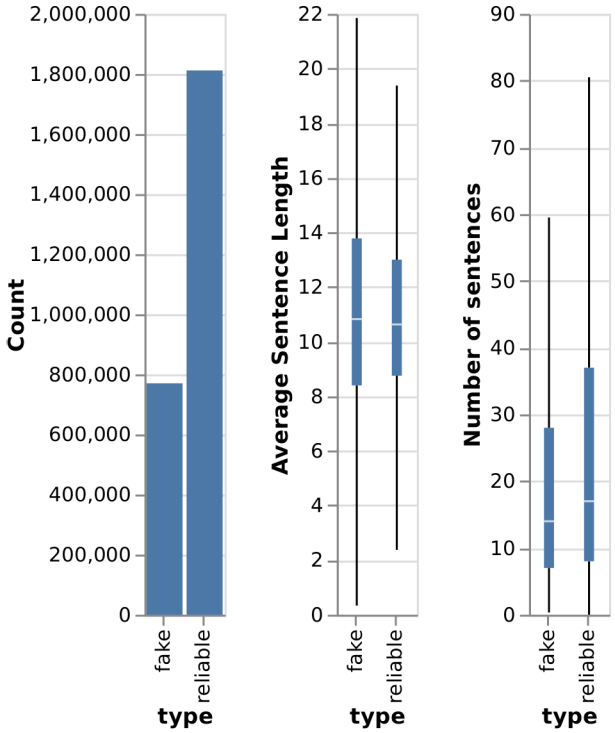



Figure 2.5: Summary statistics for not downsampled fake and reliable news.

## 2.4 Visualisation with t-SNE

In order to visualize the data it is needed to transform text in a numerical way and eventualy reduce the dimension in order to to allow it to be plotted on a 2D or 3D plot. Here TF-IDF (term frequency, inverse document frequency[5, 31]) is used. How it works will be details later on. This produce a sparse matrix with each document being represented as an array, each value of the array being a value for one term. That is, the more term in the corpus, the longer the array becomes. For example, a corpus of 10.000 text with 20.000 uniques words would be represented as a $10000 \times 20000$ sparse matrix. As said before, plotting in 20000 dimension is not possible. In order to do so, the number of dimension needs to be reduced. Here, principal component analysis before t-SNE[32] will be used together.

**What is t-SNE?** It stands for **t-distributed stochastic neighbor embedding**. The goal of this method is that two points to are closed in $\mathcal{R}^Q$ should be close in $\mathcal{R}^S, S \ll Q$. In order to do so, the algorithm starts by fitting a probability distribution on the input points. It starts by computing

$$p_{i|j} = \frac{\frac{exp(-||x_i - x_j||^2)}{2\sigma_i^2}}{\sum_{k \neq i} \frac{exp(-||x_i - x_k||^2)}{2\sigma_i^2}}$$

Each of the $y_i, i \in 0 \dots N$ being a point of dimension $Q$.
Then

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N}$$

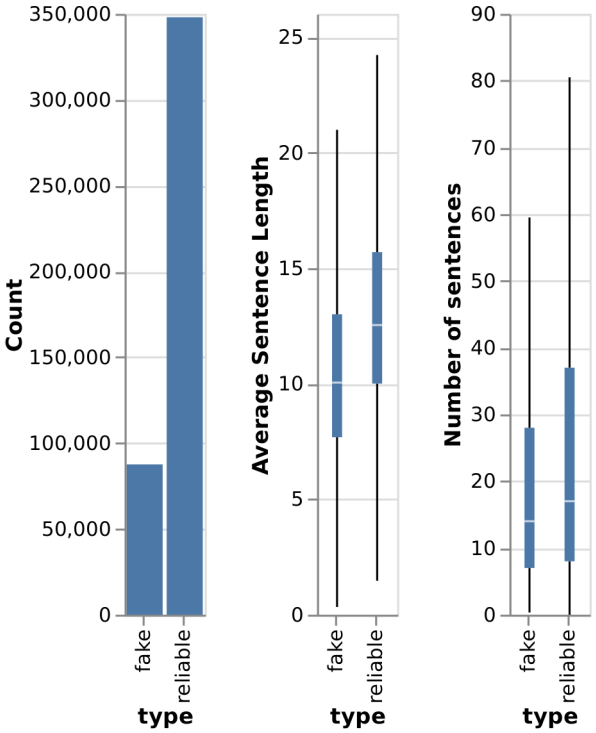



Figure 2.6: Summary statistics for downsampled dataset on fake and reliable news

is computed.

The probability distribution for the low density map is given by

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - x_l||^2)^{-1}}$$

And in order to have a low dimension probability distribution as close as possible to the high dimension one, it minimize the KullbackLeibler divergence of the two distributions.

$$KL(P||Q) = \sum_{i \neq j} p_{ij} * \log(\frac{p_{ij}}{q_{ij}})$$

The minimization of KL divergence is acheived by gradient descent.
The documentation of the module of scikit-learn[33] used for computing this value recomand to apply PCA first in order to reduce the original dimension and speed-up computation. The result of these computation can be seen at **Figure 2.8**. These 500 firsts PCA components explains around 47% of the total variance. The figure shows that there is no clear clustering of the classes.
Increasing the number of PCA components to 1750 gives the results at **Figure 2.9** and does not shows more clustering, even if it explains 75% of the variance. This shows that classifiying the dots might not be easy, but it should be remineded that it is a dimensionality reduction and that there is a loss of information. Some of the original data dimension can have a better separation of the classes.

It is not possible to use t-SNE on the *Fake News Corpus* because the algorithm is quadratic with respect to the number of samples. Which make it impossible to compute

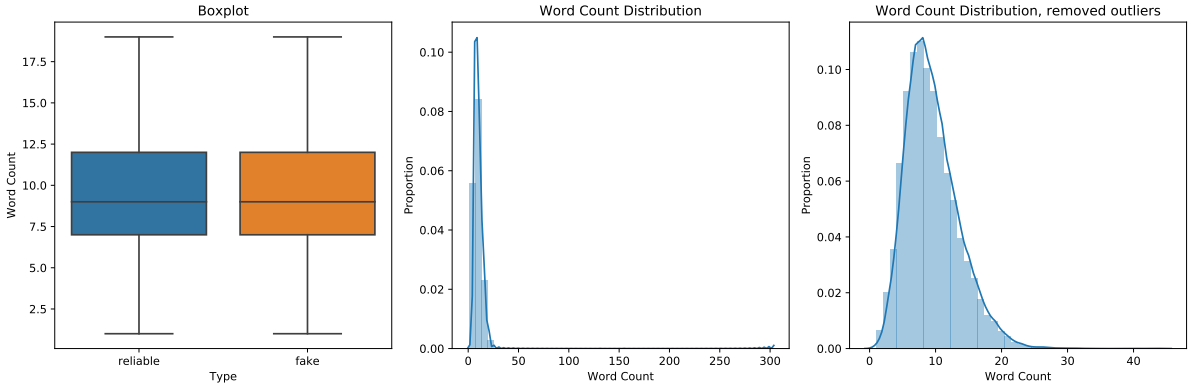



Figure 2.7: Number of words distributions for liar-liar dataset. On the first and the third plots, a few outliers with length greater than 50 have been removed in order to make the plots more readables.

for that corpus which is larger than the *liar-liar* corpus. But it is still possible to try to make some vizualisation using truncated singular v alues decomposition.

## 2.5 Conclusion

Data exploration have shown that there is no real statistical differences between text metadata for fake and reliable news, and thus make it not interesting for using it for classifying new texts. In addition, dimensionality reduction does not show any sign of helpfulness for the classification.

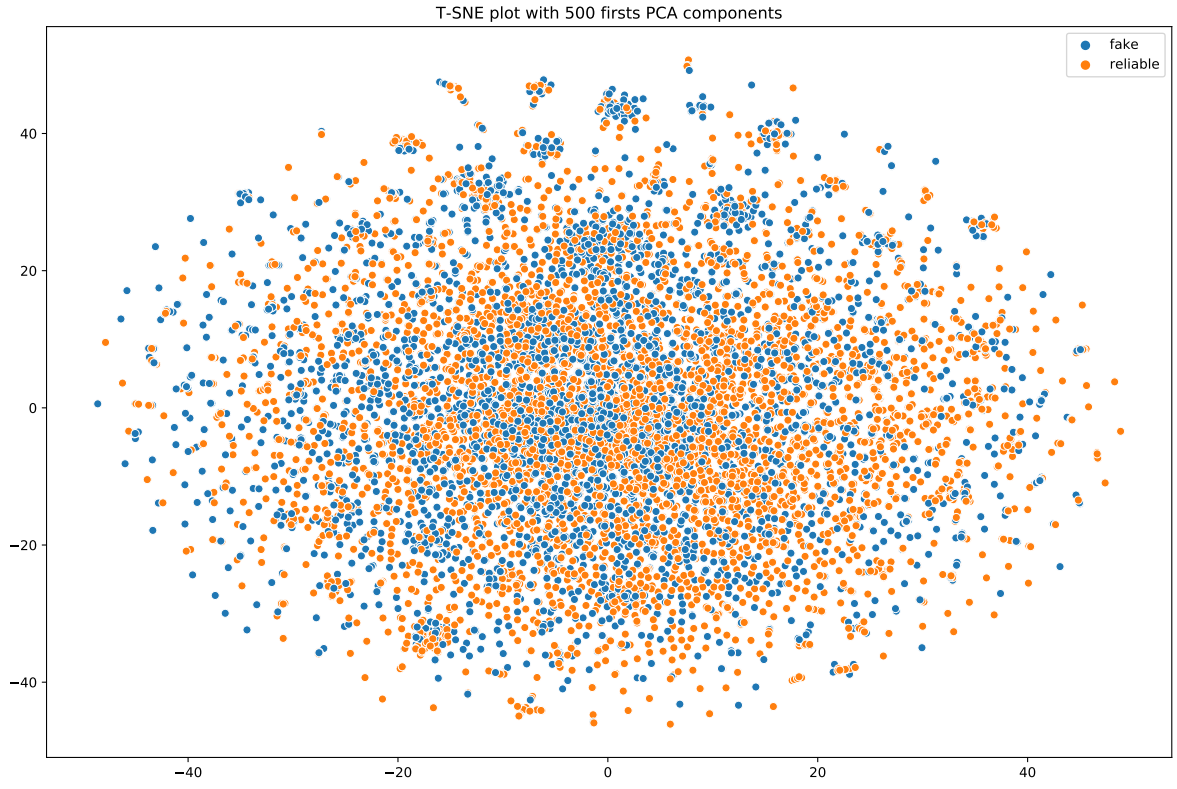



Figure 2.8: t-SNE plot for liar-liar dataset.

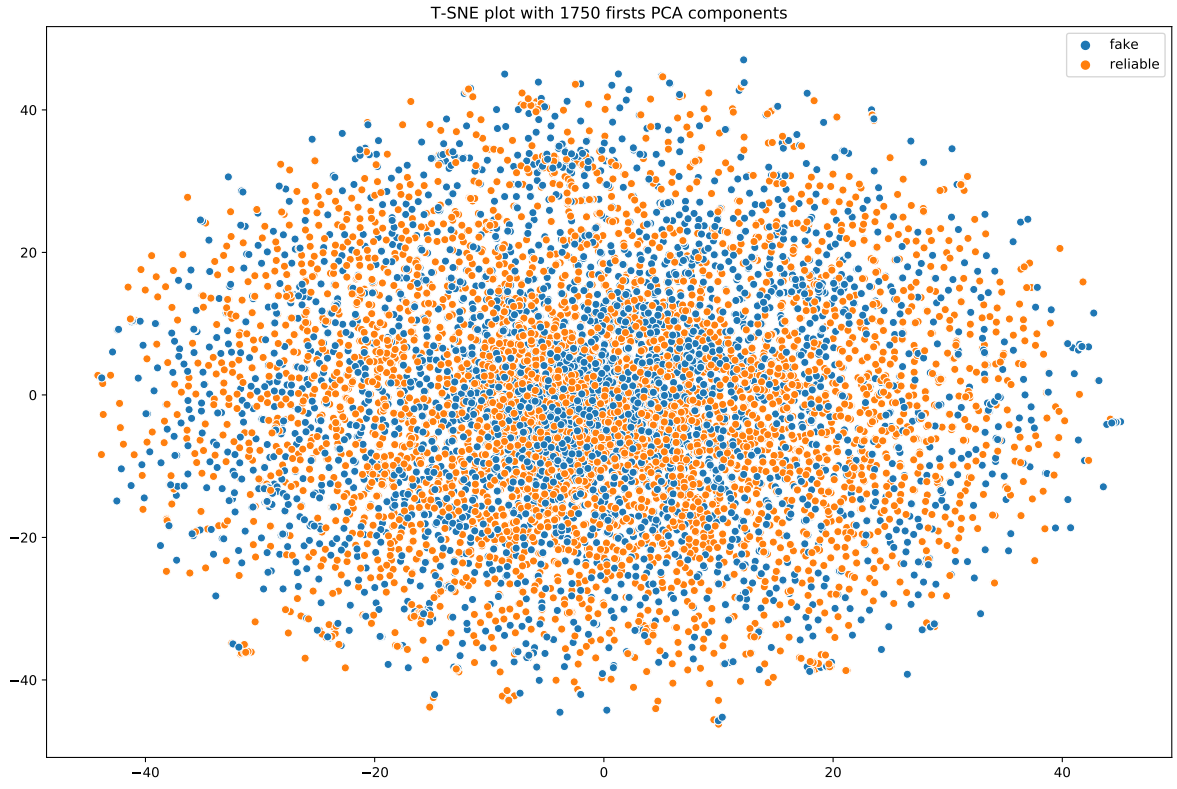



Figure 2.9: t-SNE plot for liar-liar dataset.

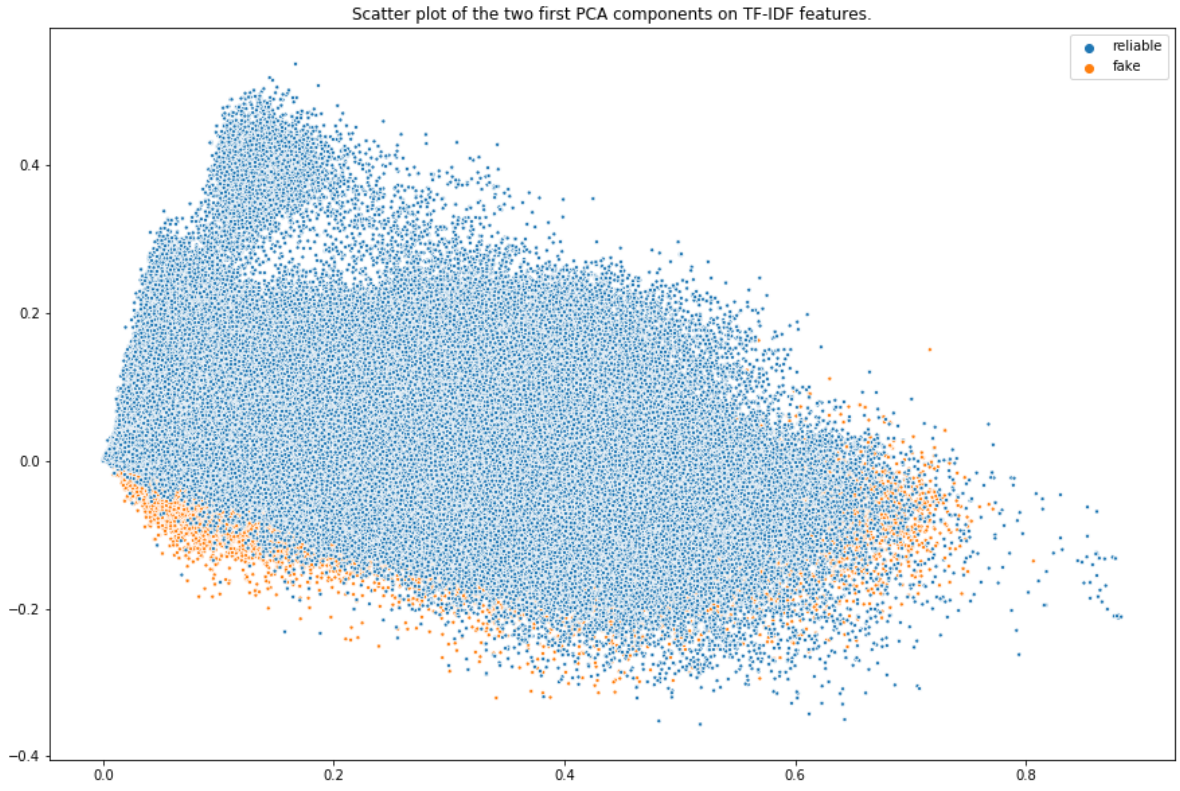



Figure 2.10: First two LSA compnoents for fake news corpus

# Chapter 3

# Machine Learning techniques

## 3.1 Introduction

In this chapter, we will focus on the more traditional methods used in natural language processing such as Nave-Bayes, decision trees, linear SVM and others. These will serve as baseline for comparing the performances of the more two advanced models that will be analyse later on: LSTM and Attention Mechanism. The first thing to do when working with text is the do words and texts embedding, indeed, in order to use machine learning algorithms on texts, a mathematical representation of these texts is required.

## 3.2 Text to vectors

As explained before, text need to be represented in a way that gives more meaningful information than a simple sequence of bits, which have additional drawbacks such that for a given word, the sequence of bits representing it depends on the coding. The first and simplest coding that come to mind is a one-hot encoding: a matrix $M$ of size number of texts $\times$ number of words where $M_{ij} = 1$ if the word $j$ is present in the text $i$ and 0 in the other case. But this is till not enougth as each word is given the same weight, no matter how often it appears in the text.

In order to overcome this problem, term-frequency might be used, that is, rather than setting $M_{ij}$ to 0 or 1 we set it to the number of time it appears in the text.

It is possible to use even better text embedding. It is called term-frequency, inverse document frequency. The main idea is that a word that appears often in all the documents is not helpful in order to classify the documents. For example, if the task is to classify books of biology and physics, words atom, cell or light are more useful than today or tomorrow.

In order to compute tf-idf, it is separted in two part, the first one being term-frequency and the second one inverse document frequency. We have that

$$tf_{ij} = \#(W_j | W_j \in D_i) \tag{3.1}$$

That is, $tf_{ij}$ is the number of time the word j appears in the document i.
Secondly, we have that

$$idf_j = \log(\frac{\#D}{\#(D_i | W_j \in D_i)})$$

This is the log of the total number of documents, over the number of documents that contains the word j. Finnaly, the value tfidf value is computed by

$$tf - idf_{ij} = tf_{ij} * idf_j \tag{3.2}$$

This the text embedding methods that will be used in this section.

## 3.3 Methodology

All the methods presented will be tested in tree differents ways:

- On the liar-liar dataset
- On the fake corpus dataset, excluding the news from *beforeitsnews.com* and *nytimes.com*
- On the fake corpus dataset, including the two pervious domains.

To be more precise, in the first case, the models will be trained on a training set, tunned using validation set and finally tested using test set. In the second case, the same methodology will be used, the dataset have been split be choosing 60% of the text from each domain for training, and 20% for validation and testing. This way have splitting have been choosen because of the uneven representation of each domain in the dataset in order to ensure representation of all the domains in the tree subsets.
In the last case, the model is trained on all the news but the two domain previously cited and tested on this two domains.

### 3.3.1 Evaluation metrics

In order to evaluate each models, multiple evaluation metrics have been used. There are recall, precision and f1-score. It is needed to use multiple metrics because they don't all account for the same values. For instance, it is possible to have a model with a recall of 1 that behave extremely bad because it simply classify all the inputs in the same single class.
Remember that precision is defined as

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

Which means that we can have two different precision, depending on which classes is considered as being positive. This is the proportion of correctly classified positive element over the number of element classified as positive. It is equals to 1 when there is no false positive, but it does not means that all the positive elements are correctly classified as it might be some false negative. The recall helps to solve this problems.
It is defined as

$$recall = \frac{TP}{TP + FN} \tag{3.4}$$

The f1-score combines the recall and the precision. It is defined by

$$f1 - score = \frac{2 * precision * recall}{precision + recall} \tag{3.5}$$

It is also possible to look at the weighted average of all these values. For instance, it is possible to compute un global recall be average the recall for both class by the respective class ratio.

Finally, raw output can be used by looking at the confusion matrix.

The first parameters to tune is the max number of features used by tf-idf. This is the maximum number of words that will be kept to create the text encoding. The words that are kept are the most frequent words.

## 3.4 Models

Four models have been used in order to classify texts represented as a TF-IDF matrix. These are Multinomial Nave-Bayes, Linear SVM, Ridge Classifier and Decision Tree.

### 3.4.1 Naïve-Bayes[6]

The basic idea of Nave-Bayes model is that all features are independent of each other. This is a particulary strong hypotheses in the case of text classification because it suppose that words are not related to each others. But it know to work well given this hypotheses. Given an element of class y and vector of feature $\mathbf{X} = (x_1, ..., x_n)$. The probability of the class given that vector is defined as

$$P(y|\mathbf{X}) = \frac{P(y) * P(\mathbf{X}|y)}{P(\mathbf{X})} \tag{3.6}$$

Thanks to the assumption of conditional independance we have that

$$P(x_i|y, x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) = P(x_i|y) \tag{3.7}$$

Using bayes rules we have that

$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n)} \tag{3.8}$$

Because $P(x_1, ..., x_n)$ is constant, we have the classification rule

$$\hat{y} = \underset{y}{argmax} P(y) \prod_{i=1}^{n} P(x_i|y) \tag{3.9}$$

### 3.4.2 Linear SVM

Linear SVM is a method for large linear classification. Given pairs of features-label $(\mathbf{x_i}, y_i), y_i \in \{-1, 1\}$, it solve the following unconstrained optimization problem.

$$\underset{w}{min} \frac{1}{2}\mathbf{w^T w} + \mathbf{C}\sum_{i=1}^{l} \xi(\mathbf{w}; \mathbf{x_i}, y_u) \tag{3.10}$$

Where $\xi$ is a loss function, in this case L2 loss function have been used, and $\mathbf{C} > 0$ a pennality parameter.
Class of new examples are assigned by looking at the value of $\mathbf{w^T w}$. The class 1 is assigned if $\mathbf{w^T w} \geq 0$ and the class $-1$ if $\mathbf{w^T w} < 0$.

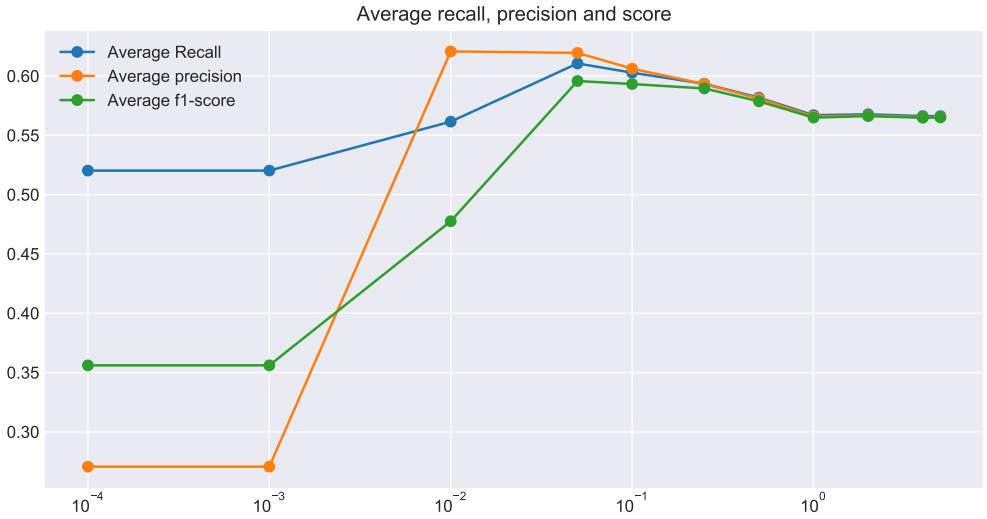



Figure 3.1: Tuning linearSVC parameters

### 3.4.3 Decision Tree

### 3.4.4 Ridge Classifier

Ridge classifier works the same way as ridge regression. It states the problem as a minimization of the sum of square errors with penalization. It can be expressed as in **Equation 3.11**.

$$\min_{w} ||Xw - y||_2^2 + \alpha ||w||_2^2 \tag{3.11}$$

The predicted class if positive if Xw is positive and negative otherwise.

## 3.5 Models on liar-liar dataset

### 3.5.1 Linear SVC

In the case of linear SVC there is one parameters to tune up, which is the penality parameters for the error term. **Figure 3.1** show the tree main metrics with respect to the panlity parameter. This show that a parameters of around 0.1 is the best one.

### 3.5.2 Decision Tree

With decision tree, it is possible to reduce overfitting by pruning the tree. It is possible to do preprunning or post pruning. Pre pre preuning means that a stoping criterion is used to stop tree growth earlier and post pruning cut the tree once it have been fully grown. It this case pre pruning is done by limiting the maximum depth of the tree.
**Figure 3.2** shows metrics values for different depths. It seems that tree of depth 1000 are the best ones.

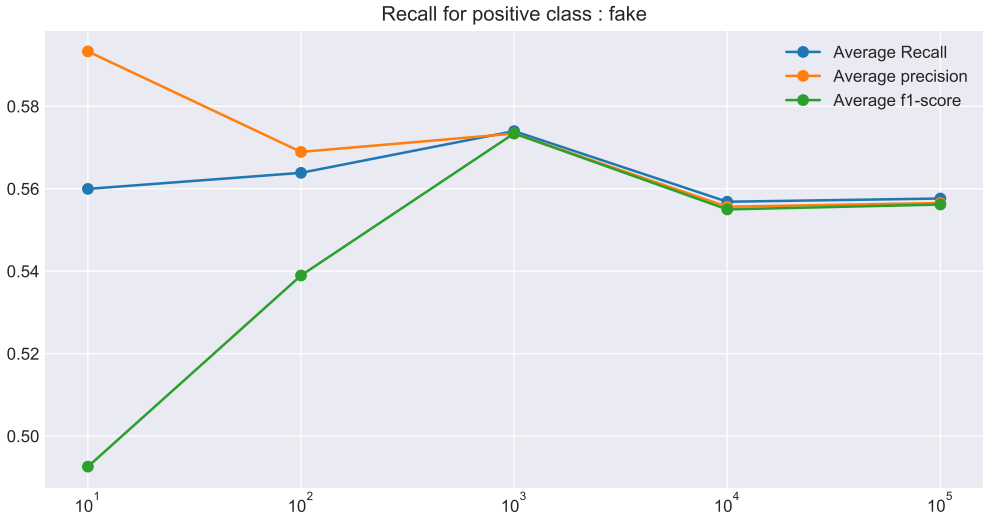



Figure 3.2: Tuning decision tree parameters

### 3.5.3 Ridge Classifier

With the ridge classifier model it is also possible to tweak the pennality value of the optimization problem. At **Figure 3.3** we can see that the optimal parameter is around 10 or 20, depending of the metrics that we want to maximize. Later on, the value of 10 will be chose as a comprimse between precision and recall. It is the value that maximize the f1-score.

### 3.5.4 Max Feature Number

Starting with the maximum number of features, the precision of each model can be analyzed when limiting the maxium number of words. The results for each model can be seen at **Figure 3.4, 3.5** and **3.6**. Shows that depending on the metrics we want to optimize it is better to choose different parameters. For instance, in order to maximize F1-score, it is better to use a maximum number of features of 1000.

The results are slightly different if the goal is to optimize the precision because if the best value stay the same for Linear SVM and Ridge Classifier, the Nave-Bayes works better when using the maxium number of features and it goes the same way for recall. Based on **Figure 3.4** we can say that when it comes to precision and recall, Nave-Bayes is the one that perform the best.

Row results for max features selection are available at **Appendix A**. It goes differently when we focus on a single class. For example, the precision for fake detection is at its maxium for Linear SVM and Ridge Classifier when only 10 features are used. But at the same time, it is at its minium for reliable class. Its shows that when trying to optimze the overall model and not only for a single class, it is better to look at the weighted average than at the value for a single clas. But it is still important to look at the metrics for a single class because it indicate how it behave for this class. For instance, in the case of automatic fake news detection, it is important to minimize the number of reliable news

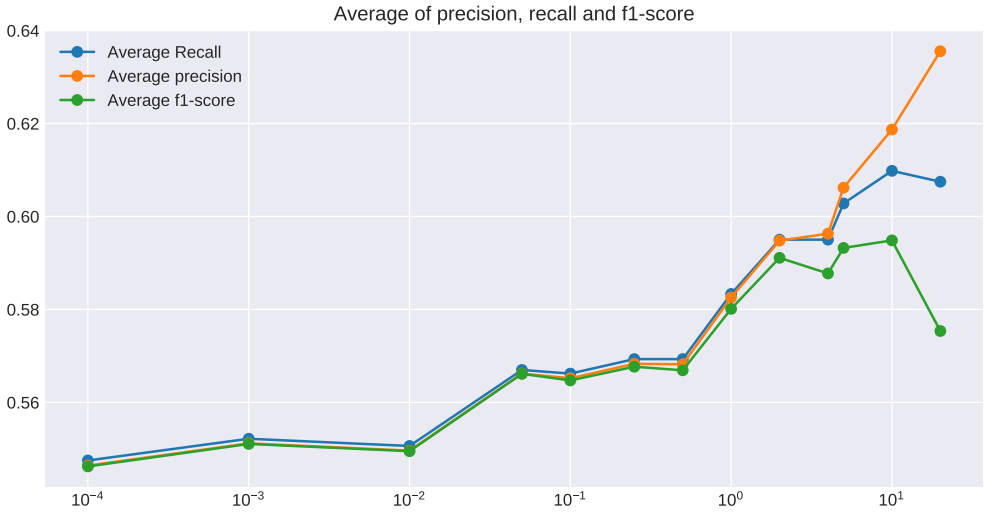



Figure 3.3: Average metrics for ridge classifier with respect to the penality parameter.

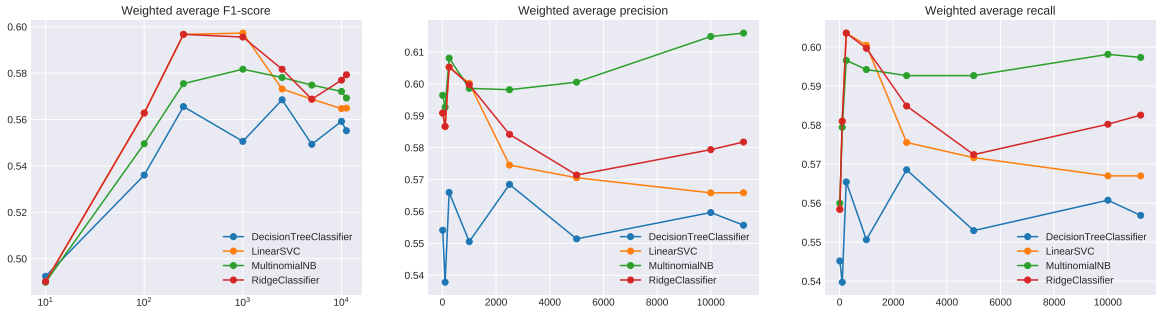



Figure 3.4: Weigted average of f1-score, precision and recall of each classes.

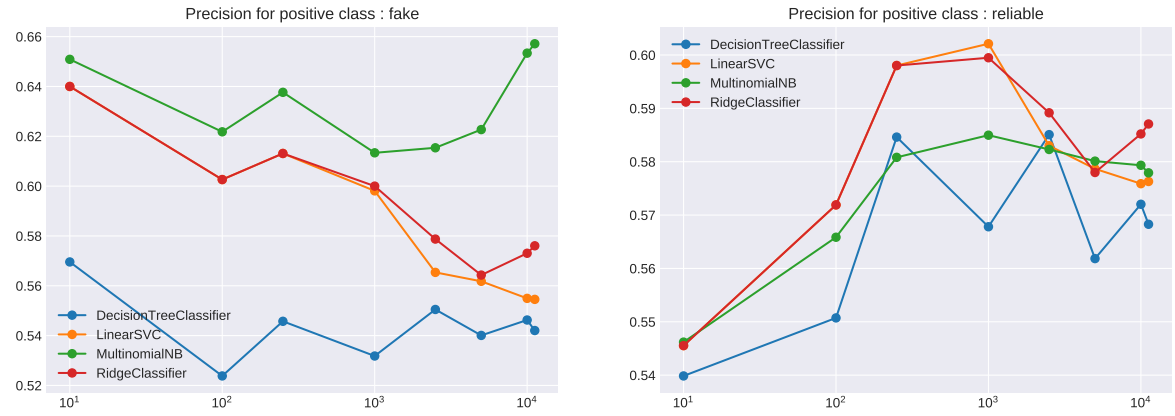



Figure 3.5: Precision of the model for each classes, the x axes is log scale of the number of features

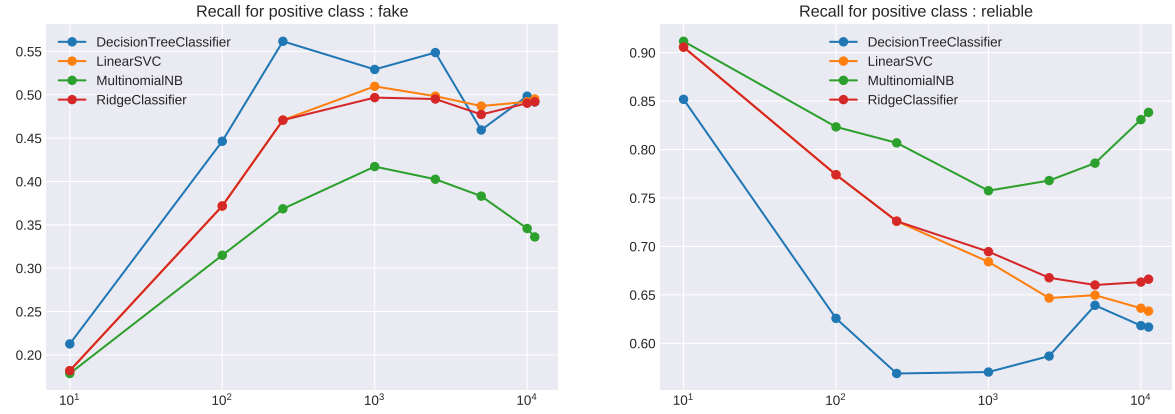



Figure 3.6: Precision of the model for each classes, the x axes is log scale of the number of features

missclassified in order to avoid what could be called censorship.

## 3.6 Models on fake corpus dataset

### 3.6.1 SMOTE: Synthetic Minority Over-sampling Technique[34]

As it have been show in **Chapter 2**, the fake news corpus is inbalanced. Synthetic minority over-sampling is a technique that allows to generate fake samples from the minor class. It works by randomly choosing one or many neirest neigbhor in the minority class. For instance, if the algorithm is set to use 5 nearest neigbours, for each samples it will choose one of its nearest neigbours, and generate a new sample on the segment joining the sample and its neigbour.
**Algorithm 1** and **2** shows how it works. The first one compute the k-nearest neighbors and the second one compute a new element by randomly choosing one of this neighbors.

**Data:** k = Number of nearest neighbors
**Data:** T = number of minority class samples
**for** $i \leftarrow 1...T$ **do**
  Compute k-nearest neighbors of sample i;
  Populate(knn, i);
**end**

**Algorithm 1:** SMOTE

**Data:** knn = the k nearest neighbour of sample i
**Data:** s = ith sample
nn = random_choice(knn);
newSample = s + rand(0, 1) * (nn - s);

**Algorithm 2:** Populate

### 3.6.2 Results without using SMOTE

#### Hyperparameters tuning

As for the models trained on the **liar-liar corpus**, hyper-parameters can be optimized the same way. The average metrics for each models with respect to there paramters are shown at **Figure 3.7, 3.8** and **3.9**.

The optimal parameter for the ridge classifier is clearly 1. As well as for the decision tree trained on **liar-liar** dataset, the optimal maximum depth is of 1000. And finaly, the optimal value for the penality parameter of the svm is also 1.

By looking at **Figure 3.5, 3.6** and **3.4** we can find optimal parameters for the number of features used in TF-IDF. It shows that linear svm and ridge classifier are the ones that perform the best, having an average precision of sligtly more than 94% for the linear svm and 94% for the ridge classifier. They acheive these performances from $50,000$ features and does not decrease. On the other hand, Nave-Bayes reches a pike at $100,000$ features and greatly decrease afterward.

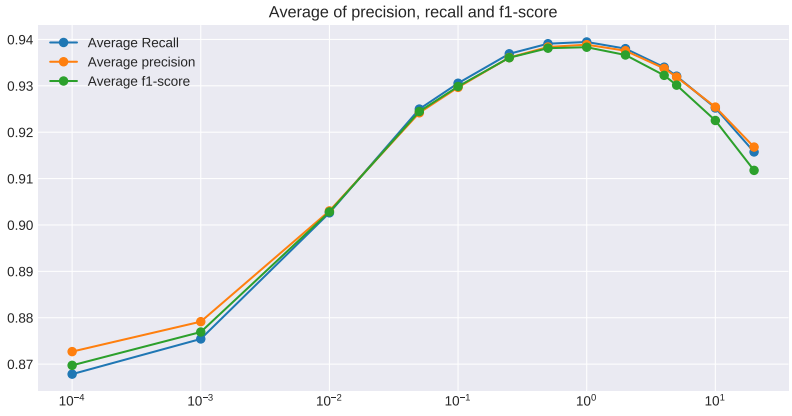



Figure 3.7: Metrics value with respect to the penality paramter for ridge classifier

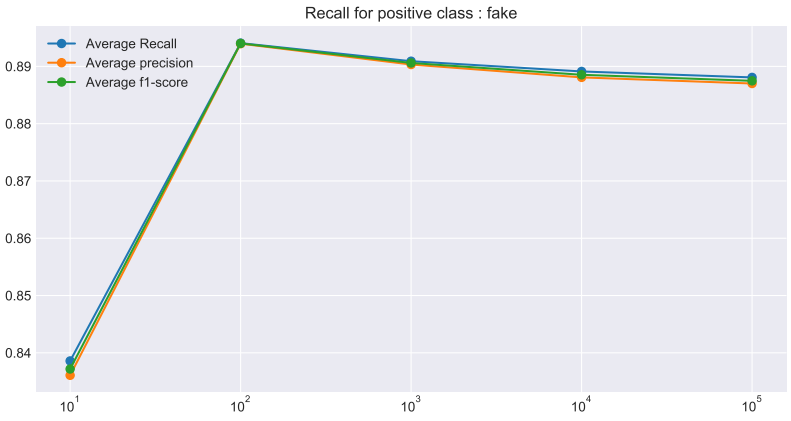



Figure 3.8: Optimal depth of decision tree.


Figure 3.9: Optimal penality parameters for linear svm

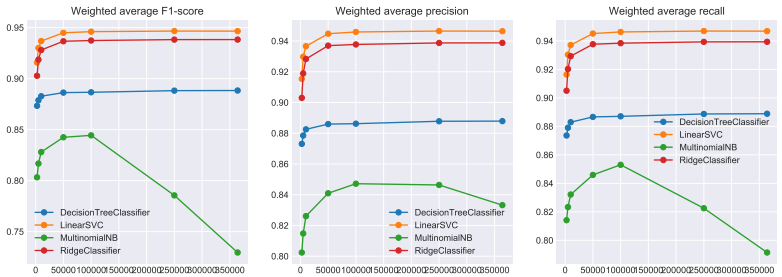



Figure 3.10: Average recall, precision and f1-score wti respect to the maxiumum number of features.


Figure 3.11: Precision for fake and reliable class for each model with respect to the maxiumum number of features

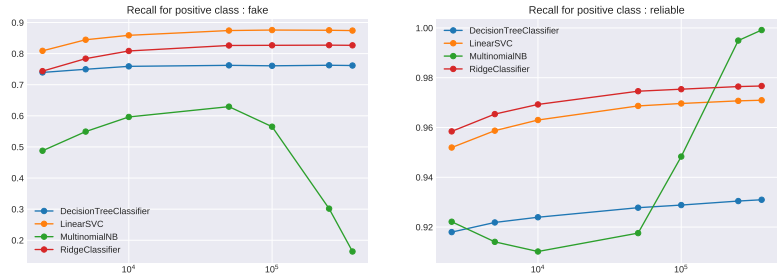



Figure 3.12: Recall for fake and reliable class for each model with respect to the maxiumum number of features

**Figure 3.4** shows why it is important to look at all the metrics, because Nave-Bayes reaches a recall of 1 for the reliable class and close to 0 for the fake class, which means that almost all the text are classified as reliable. This can be verified by looking at **Figure 3.13**, only a small proportion of true fake are actualy classified as it.

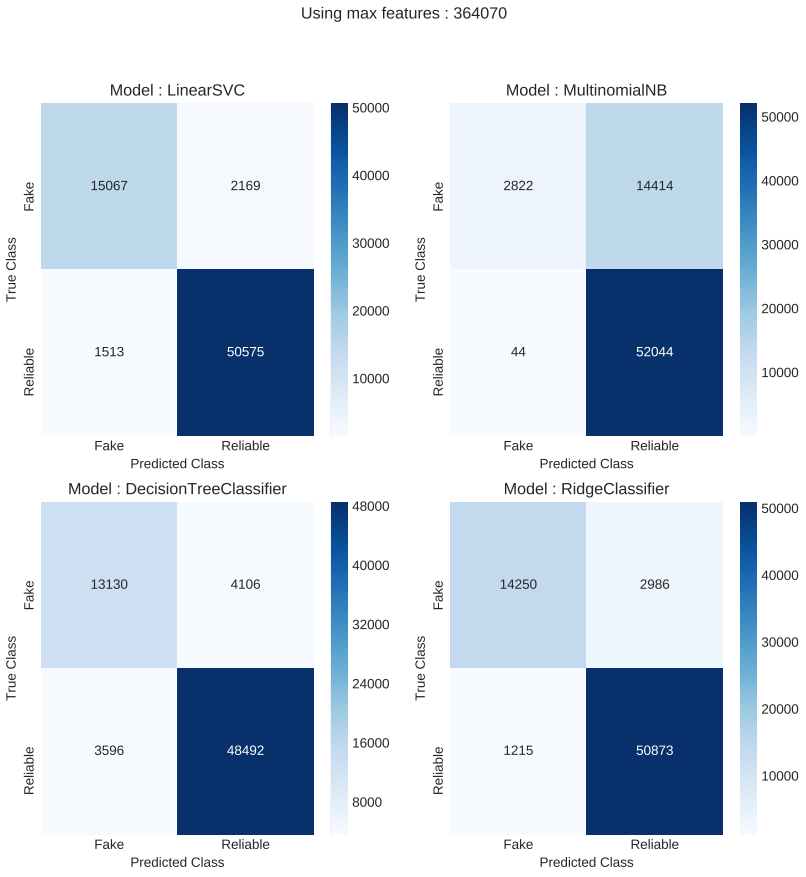



Figure 3.13: Confusion matrix for each model using 364,070 features

# Chapter 4

# Attention Mechanism

## 4.1 Introduction

In this section, we will focus on the deep learning models, the first one being a bidirectional LSTM and the second one an attention layer is added to this LSTM. But it is need to use an other text embedding in order to work with LSTM. Indeed, tf-idf create a sparse matrix with each row corresponding to a value for a given word. This means that the order of the words are lost. In order to solve this, word2vec[35] is used. It allows to match words to continuous vectors of a given size with interesting properties. An other methods, which consist in making word embedding as tuning parameters will be used.

## 4.2 Text to Vectors

### 4.2.1 Word2Vec

Word2Vec comes in two fashions: continuous bag of word (CBOW) and skip-gram. It is originaly designed to predic a word given a context. For instance, given two previous words and the two next words, which word is the most likely to take place between them. But it appears that the hidden representation of these words works well as word embedding and have very interesing properties such that words with similar meaning have similar vector representation. It is also possible to perform arithmetic that captures informations such as singular, plural or even capital and countries. For example, we have that $dog - dogs \approx cat - cats$ but also $Paris - France \approx Berlin - Germany$.

It is possible to visualize these relasionships by using t-SNE for projecting high dimentions word vectors in 2D space. The results of various relashionships can be see at **Figure 4.1**.

#### How does it works?

As the orignal authors did not inteded this kind of results, Xin Rong[36] did a good job explaing how it works.
Let V be the size of the vocabulary and that there is only one word in the CBOW model, it give **Figure 4.2** model.
Each words is encoded as a one-hot vector of size V. That means that it is a sparse vector full of zeros except for the position assigned to that word which is one. The hidden layers

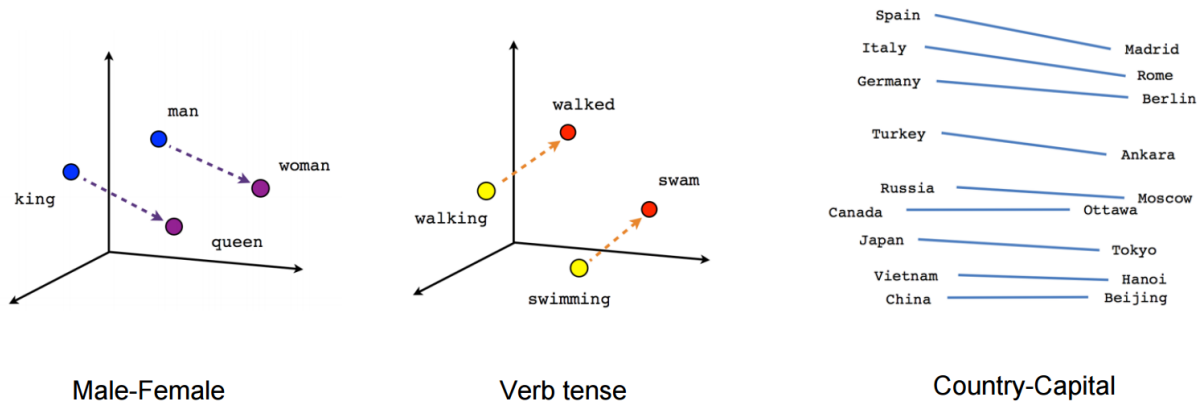



Figure 4.1: Relashionships between different words with t-SNE dimentionality reduciton.

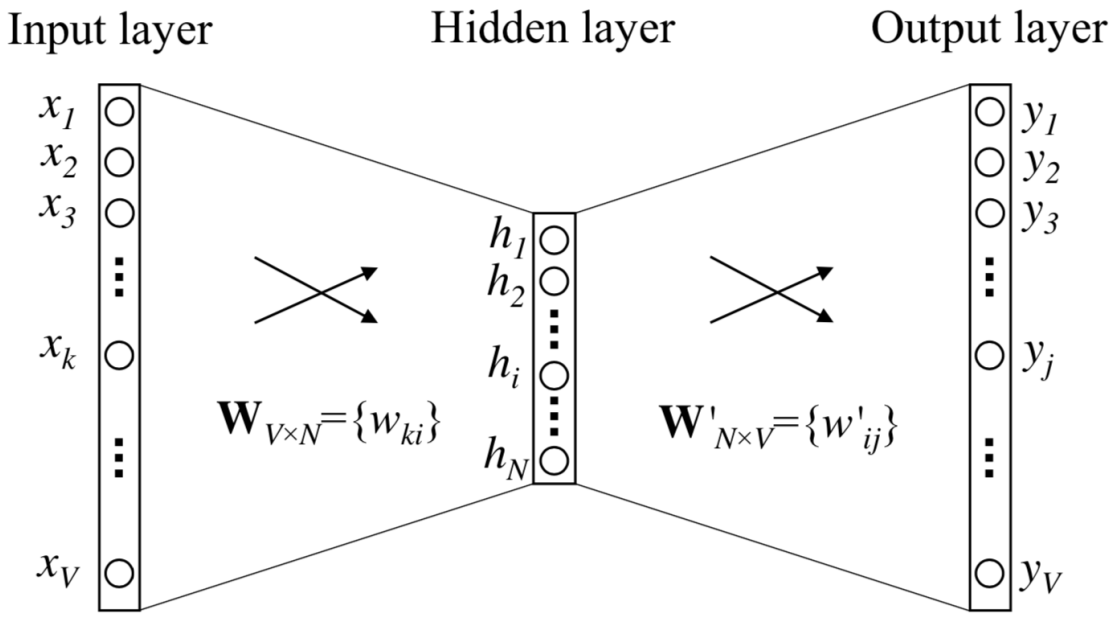



Figure 4.2: A simple CBOW model with only one word in the context

is computed as

$$\mathbf{h} = \mathbf{W^T x} \tag{4.1}$$

Where $\mathbf{W^{V \times N}}$ is the weighte matrix to optimize over.
The output layer values are computed as

$$\mathbf{Y} = \mathbf{W'^T h} \tag{4.2}$$

As before $\mathbf{W'^{N \times V}}$ is also a weight matrix to optimize. The loss can be computed as softmax cross entropy.

It is also possible to make the oposite: predicting the context given a single input word. This is the skip-gram model. In this case the loss become **Equation 4.3**.

$$E = -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \sum_{j'=1}^{V} \exp(u_{j'}) \tag{4.3}$$

$j_c^*$ is the index of the cth output context word and $u_{j_c^*}$ is the score of the jth word in the vocabulary for the cth context word. Finly, the embedding that is used are the value of the hidden layers produced for a given word.

## 4.3 LSTM

LSTM or Long Short Term Memory[7] is a kind of reccurent neural network that fits well to temporal or sequential input such as texts. A RNN is a type of neural network where the hidden state is fed in a loop with the sequential inputs. There are usualy shown as unrolled version of it (**Figure 4.4**). Each of the $X_i$ being one value in the sequence.

Recurrent Neural Networks does not works very well with long-term dependencies, that is why LSTM have been introduced. It is made of an input gate, an output gate and a forget gate that are combined in **Equation 4.4**.

$$f_t = \sigma_g(\mathbf{W}_f x_t + \mathbf{U}_f h_{t-1} + b_f) \tag{4.4}$$
$$i_t = \sigma_g(\mathbf{W}_i x_t + \mathbf{U}_i h_{t-1} + b_i) \tag{4.5}$$
$$o_t = \sigma_g(\mathbf{W}_o x_t + \mathbf{U}_o h_{t-1} + b_o) \tag{4.6}$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(\mathbf{W}_c x_t + \mathbf{U}_c h_{t-1} + b_c) \tag{4.7}$$
$$h_t = o_t \circ \sigma_h(c_t) \tag{4.8}$$

**Figure 4.5** shows how it works.

## 4.4 Attention Mechanism

## 4.5 Results

### 4.5.1 Methodology

In order to train the models and perform hyper parameters optimization grid search have been used when it was possibile (on the liar-liar dataset) and knwoleadge aquiered there

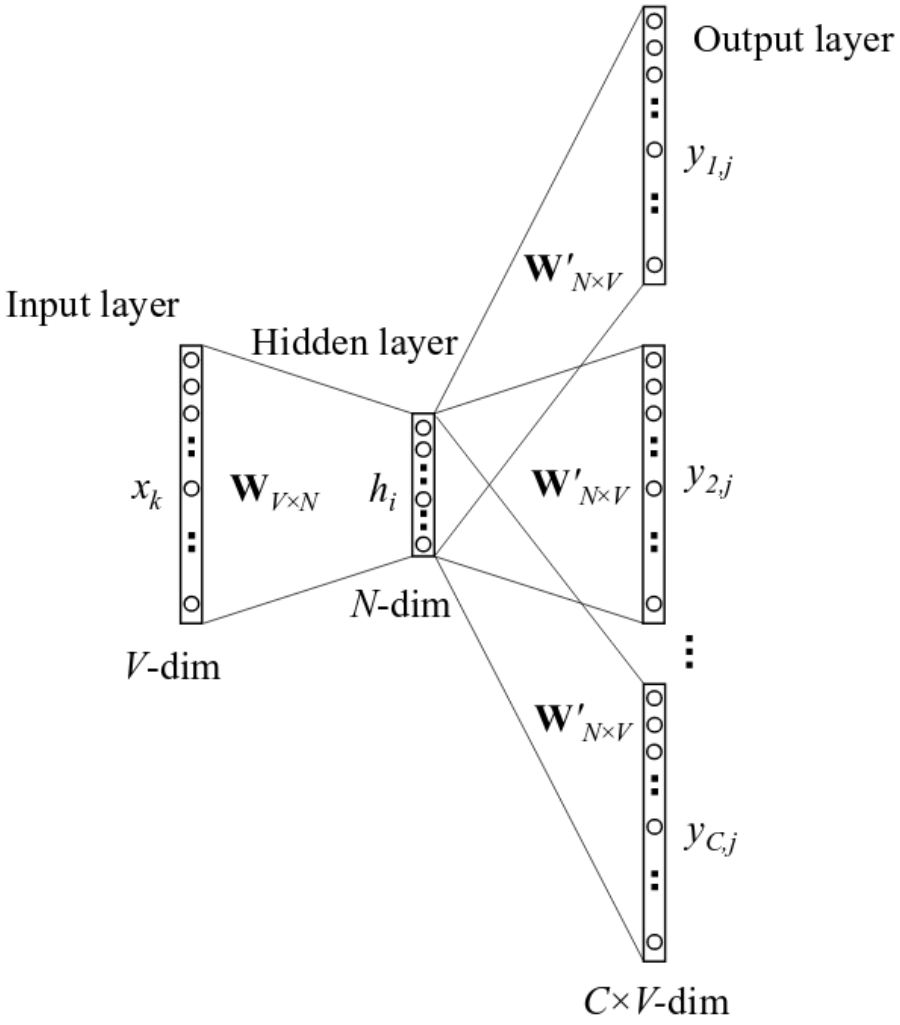



Figure 4.3: Skip-gram model with multiple outputs.

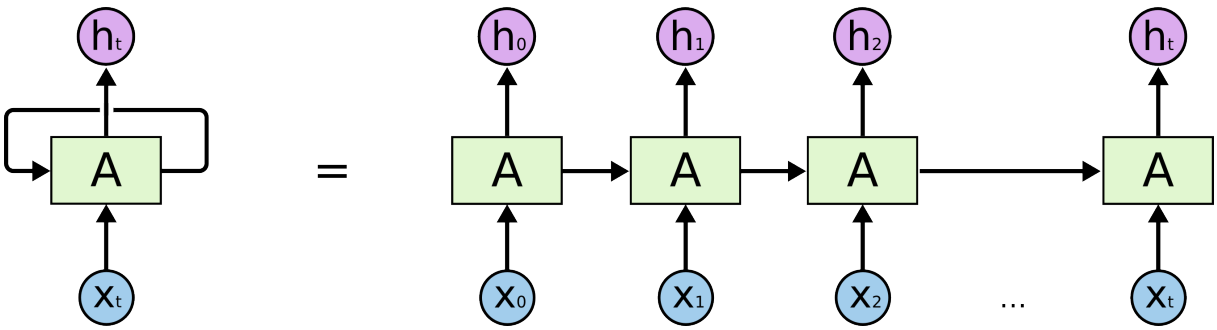



Figure 4.4: Unrolled RNN (Understanding LSTM Networks, `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`)

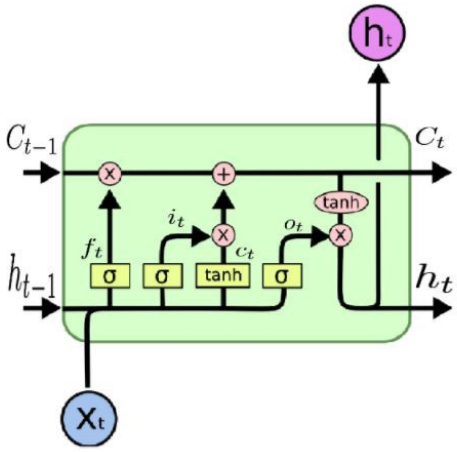



Figure 4.5: LSTM gates,
`https://hackernoon.com/understanding-architecture-of-lstm-cell-from-scratch-with-code-8da40f0b71f4)`

have been used in order to tune parameters for the networks on the **Fake News Corpus**. In addition, in order to find the bests parameters among all tested with gird search, for each metrics, the training epochs having the highest validation value for that metrics have been choosen.

As SMOTE cannot be used on the **Fake News Corpus** dues to the size of the corpus, in order to rebalance the dataset the minority class have been over sampled by feeding multiple times the same input by looping through them.

### 4.5.2 Liar-Liar dataset results

As explained earlier, both models have been train using different embedding: the first one being pre-trained word2vec vectors of size 300 and the second one beging a tunable parameters with different embedding size.

**Using word2vec embedding**

# Bibliography


[1] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. In *Journal of Economic Perspective*, volume 31, 2017.

[2] Jeffrey Gottfried and Elisa Shearer. *News Use Across Social Medial Platforms 2016*. Pew Research Center, 2016.

[3] Craig Silverman and Lawrence Alexander. How teens in the balkans are duping trump supporters with fake news. *Buzzfeed News*, 3, 2016.

[4] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, 2008.

[5] Stephen Robertson. Understanding inverse document frequency: On theoretical arguments for idf, 2004.

[6] Harry Zhang. The Optimality of Naive Bayes. page 6.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[8] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany, August 2016. Association for Computational Linguistics.

[9] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36, 2017.

[10] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[11] Natali Ruchansky, Sungyong Seo, and Yan Liu. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 797–806. ACM, 2017.

[12] Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. Fake news detection through multi-perspective speaker profiles. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 252–256, 2017.

[13] Eugenio Tacchini, Gabriele Ballarin, Marco L. Della Vedova, Stefano Moret, and Luca de Alfaro. Some like it hoax: Automated fake news detection in social networks.

[14] James Thorne, Mingjie Chen, Giorgos Myrianthous, Jiashu Pu, Xiaoxuan Wang, and Andreas Vlachos. Fake news stance detection using stacked ensemble of classifiers. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*, pages 80–83, 2017.

[15] Mykhailo Granik and Volodymyr Mesyura. Fake news detection using naive bayes classifier. In *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, pages 900–903. IEEE, 2017.

[16] Yang Yang, Lei Zheng, Jiawei Zhang, Qingcai Cui, Zhoujun Li, and Philip S. Yu. Ti-cnn: Convolutional neural networks for fake news detection.

[17] Yaqing Wang, Fenglong Ma, Zhiwei Jin, Ye Yuan, Guangxu Xun, Kishlay Jha, Lu Su, and Jing Gao. Eann: Event adversarial neural networks for multi-modal fake news detection. In *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining*, pages 849–857. ACM, 2018.

[18] Julio CS Reis, André Correia, Fabrício Murai, Adriano Veloso, Fabrício Benevenuto, and Erik Cambria. Supervised learning for fake news detection. *IEEE Intelligent Systems*, 34(2):76–81, 2019.

[19] Vernica Prez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. Automatic detection of fake news.

[20] James W Pennebaker, Martha E Francis, and Roger J Booth. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001, 2001.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[22] Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. Independently recurrent neural network (indrnn): Building a longer and deeper rnn.

[23] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical Attention Networks for Document Classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California, 2016. Association for Computational Linguistics.

[24] Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. Adversarial Training Methods for Semi-Supervised Text Classification. *arXiv:1605.07725 [cs, stat]*, May 2016. arXiv: 1605.07725.

[25] Yoon Kim. Convolutional Neural Networks for Sentence Classification. *arXiv:1408.5882 [cs]*, August 2014. arXiv: 1408.5882.

[26] Kamran Kowsari, Mojtaba Heidarysafa, Donald E. Brown, Kiana Jafari Meimandi, and Laura E. Barnes. RMDL: Random Multimodel Deep Learning for Classification. *Proceedings of the 2nd International Conference on Information System and Data Mining - ICISDM '18*, pages 19–28, 2018. arXiv: 1805.01890.

[27] Maciej Szpakowski. Fake news corpus. `https://github.com/several27/FakeNewsCorpus`. Accessed: 2018-10.

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

[29] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[30] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[31] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 2004.

[32] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, (9):2579–2605, 2008.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[34] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique.

[35] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space.

[36] Xin Rong. word2vec parameter learning explained.

# Appendix A

## A.1 TF-IDF max features row results on liar-liar corpus

### A.1.1 Weigted Average Metrics

| | model | max_features | recall | precision | f1 |
|---|---|---|---|---|---|
| 0 | LinearSVC | 10 | 0.558411 | 0.590855 | 0.490108 |
| 1 | MultinomialNB | 10 | 0.559969 | 0.596418 | 0.489844 |
| 2 | DecisionTreeClassifier | 10 | 0.545171 | 0.554105 | 0.492387 |
| 3 | RidgeClassifier | 10 | 0.558411 | 0.590855 | 0.490108 |
| 4 | LinearSVC | 100 | 0.580997 | 0.586645 | 0.562808 |
| 5 | MultinomialNB | 100 | 0.579439 | 0.592686 | 0.549533 |
| 6 | DecisionTreeClassifier | 100 | 0.539720 | 0.537812 | 0.536041 |
| 7 | RidgeClassifier | 100 | 0.580997 | 0.586645 | 0.562808 |
| 8 | LinearSVC | 250 | 0.603583 | 0.605262 | 0.596719 |
| 9 | MultinomialNB | 250 | 0.596573 | 0.608079 | 0.575478 |
| 10 | DecisionTreeClassifier | 250 | 0.565421 | 0.565966 | 0.565582 |
| 11 | RidgeClassifier | 250 | 0.603583 | 0.605262 | 0.596719 |
| 12 | LinearSVC | 1000 | 0.600467 | 0.600183 | 0.597274 |
| 13 | MultinomialNB | 1000 | 0.594237 | 0.598593 | 0.581692 |
| 14 | DecisionTreeClassifier | 1000 | 0.550623 | 0.550539 | 0.550578 |
| 15 | RidgeClassifier | 1000 | 0.599688 | 0.599731 | 0.595559 |
| 16 | LinearSVC | 2500 | 0.575545 | 0.574544 | 0.573173 |
| 17 | MultinomialNB | 2500 | 0.592679 | 0.598169 | 0.578113 |
| 18 | DecisionTreeClassifier | 2500 | 0.568536 | 0.568482 | 0.568508 |
| 19 | RidgeClassifier | 2500 | 0.584891 | 0.584169 | 0.581693 |
| 20 | LinearSVC | 5000 | 0.571651 | 0.570574 | 0.568765 |
| 21 | MultinomialNB | 5000 | 0.592679 | 0.600539 | 0.574854 |
| 22 | DecisionTreeClassifier | 5000 | 0.552960 | 0.551400 | 0.549321 |
| 23 | RidgeClassifier | 5000 | 0.572430 | 0.571418 | 0.568760 |
| 24 | LinearSVC | 10000 | 0.566978 | 0.565837 | 0.564715 |
| 25 | MultinomialNB | 10000 | 0.598131 | 0.614854 | 0.572109 |
| 26 | DecisionTreeClassifier | 10000 | 0.560748 | 0.559664 | 0.559212 |
| 27 | RidgeClassifier | 10000 | 0.580218 | 0.579376 | 0.576984 |
| 28 | LinearSVC | 11222 | 0.566978 | 0.565860 | 0.564915 |
| 29 | MultinomialNB | 11222 | 0.597352 | 0.615925 | 0.569280 |

| | | | | | |
|---|---|---|---|---|---|
| 30 | DecisionTreeClassifier | 11222 | 0.556854 | 0.555689 | 0.555171 |
| 31 | RidgeClassifier | 11222 | 0.582555 | 0.581782 | 0.579279 |

### A.1.2 Per class Metrics

| | model | max_features | type | recall | precision | f1 |
|---|---|---|---|---|---|---|
| 0 | LinearSVC | 10 | fake | 0.181818 | 0.640000 | 0.283186 |
| 1 | LinearSVC | 10 | reliable | 0.905689 | 0.545537 | 0.680923 |
| 2 | MultinomialNB | 10 | fake | 0.178571 | 0.650888 | 0.280255 |
| 3 | MultinomialNB | 10 | reliable | 0.911677 | 0.546188 | 0.683118 |
| 4 | DecisionTreeClassifier | 10 | fake | 0.212662 | 0.569565 | 0.309693 |
| 5 | DecisionTreeClassifier | 10 | reliable | 0.851796 | 0.539848 | 0.660859 |
| 6 | RidgeClassifier | 10 | fake | 0.181818 | 0.640000 | 0.283186 |
| 7 | RidgeClassifier | 10 | reliable | 0.905689 | 0.545537 | 0.680923 |
| 8 | LinearSVC | 100 | fake | 0.371753 | 0.602632 | 0.459839 |
| 9 | LinearSVC | 100 | reliable | 0.773952 | 0.571903 | 0.657761 |
| 10 | MultinomialNB | 100 | fake | 0.314935 | 0.621795 | 0.418103 |
| 11 | MultinomialNB | 100 | reliable | 0.823353 | 0.565844 | 0.670732 |
| 12 | DecisionTreeClassifier | 100 | fake | 0.446429 | 0.523810 | 0.482033 |
| 13 | DecisionTreeClassifier | 100 | reliable | 0.625749 | 0.550725 | 0.585844 |
| 14 | RidgeClassifier | 100 | fake | 0.371753 | 0.602632 | 0.459839 |
| 15 | RidgeClassifier | 100 | reliable | 0.773952 | 0.571903 | 0.657761 |
| 16 | LinearSVC | 250 | fake | 0.470779 | 0.613108 | 0.532599 |
| 17 | LinearSVC | 250 | reliable | 0.726048 | 0.598027 | 0.655849 |
| 18 | MultinomialNB | 250 | fake | 0.368506 | 0.637640 | 0.467078 |
| 19 | MultinomialNB | 250 | reliable | 0.806886 | 0.580819 | 0.675439 |
| 20 | DecisionTreeClassifier | 250 | fake | 0.561688 | 0.545741 | 0.553600 |
| 21 | DecisionTreeClassifier | 250 | reliable | 0.568862 | 0.584615 | 0.576631 |
| 22 | RidgeClassifier | 250 | fake | 0.470779 | 0.613108 | 0.532599 |
| 23 | RidgeClassifier | 250 | reliable | 0.726048 | 0.598027 | 0.655849 |
| 24 | LinearSVC | 1000 | fake | 0.509740 | 0.598095 | 0.550394 |
| 25 | LinearSVC | 1000 | reliable | 0.684132 | 0.602108 | 0.640505 |
| 26 | MultinomialNB | 1000 | fake | 0.417208 | 0.613365 | 0.496618 |
| 27 | MultinomialNB | 1000 | reliable | 0.757485 | 0.584971 | 0.660144 |
| 28 | DecisionTreeClassifier | 1000 | fake | 0.529221 | 0.531811 | 0.530513 |
| 29 | DecisionTreeClassifier | 1000 | reliable | 0.570359 | 0.567809 | 0.569081 |
| 30 | RidgeClassifier | 1000 | fake | 0.496753 | 0.600000 | 0.543517 |
| 31 | RidgeClassifier | 1000 | reliable | 0.694611 | 0.599483 | 0.643551 |
| 32 | LinearSVC | 2500 | fake | 0.498377 | 0.565378 | 0.529767 |
| 33 | LinearSVC | 2500 | reliable | 0.646707 | 0.582996 | 0.613201 |
| 34 | MultinomialNB | 2500 | fake | 0.402597 | 0.615385 | 0.486752 |
| 35 | MultinomialNB | 2500 | reliable | 0.767964 | 0.582293 | 0.662363 |
| 36 | DecisionTreeClassifier | 2500 | fake | 0.548701 | 0.550489 | 0.549593 |
| 37 | DecisionTreeClassifier | 2500 | reliable | 0.586826 | 0.585075 | 0.585949 |
| 38 | RidgeClassifier | 2500 | fake | 0.495130 | 0.578748 | 0.533683 |
| 39 | RidgeClassifier | 2500 | reliable | 0.667665 | 0.589168 | 0.625965 |
| 40 | LinearSVC | 5000 | fake | 0.487013 | 0.561798 | 0.521739 |
| 41 | LinearSVC | 5000 | reliable | 0.649701 | 0.578667 | 0.612130 |
| 42 | MultinomialNB | 5000 | fake | 0.383117 | 0.622691 | 0.474372 |
| 43 | MultinomialNB | 5000 | reliable | 0.785928 | 0.580110 | 0.667514 |
| 44 | DecisionTreeClassifier | 5000 | fake | 0.459416 | 0.540076 | 0.496491 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 45 | DecisionTreeClassifier | 5000 | reliable | 0.639222 | 0.561842 | 0.598039 |
| 46 | RidgeClassifier | 5000 | fake | 0.477273 | 0.564299 | 0.517150 |
| 47 | RidgeClassifier | 5000 | reliable | 0.660180 | 0.577982 | 0.616352 |
| 48 | LinearSVC | 10000 | fake | 0.491883 | 0.554945 | 0.521515 |
| 49 | LinearSVC | 10000 | reliable | 0.636228 | 0.575881 | 0.604552 |
| 50 | MultinomialNB | 10000 | fake | 0.345779 | 0.653374 | 0.452229 |
| 51 | MultinomialNB | 10000 | reliable | 0.830838 | 0.579332 | 0.682657 |
| 52 | DecisionTreeClassifier | 10000 | fake | 0.498377 | 0.546263 | 0.521222 |
| 53 | DecisionTreeClassifier | 10000 | reliable | 0.618263 | 0.572022 | 0.594245 |
| 54 | RidgeClassifier | 10000 | fake | 0.490260 | 0.573055 | 0.528434 |
| 55 | RidgeClassifier | 10000 | reliable | 0.663174 | 0.585205 | 0.621754 |
| 56 | LinearSVC | 11222 | fake | 0.495130 | 0.554545 | 0.523156 |
| 57 | LinearSVC | 11222 | reliable | 0.633234 | 0.576294 | 0.603424 |
| 58 | MultinomialNB | 11222 | fake | 0.336039 | 0.657143 | 0.444683 |
| 59 | MultinomialNB | 11222 | reliable | 0.838323 | 0.577915 | 0.684178 |
| 60 | DecisionTreeClassifier | 11222 | fake | 0.491883 | 0.542039 | 0.515745 |
| 61 | DecisionTreeClassifier | 11222 | reliable | 0.616766 | 0.568276 | 0.591529 |
| 62 | RidgeClassifier | 11222 | fake | 0.491883 | 0.576046 | 0.530648 |
| 63 | RidgeClassifier | 11222 | reliable | 0.666168 | 0.587071 | 0.624123 |

## A.2 TF-IDF max features row results for fake news corpus without SMOTE

| | model | recall | precision | max_features | f1 |
|---|---|---|---|---|---|
| 0 | LinearSVC | 0.744278 | 0.741135 | 10000 | 0.742519 |
| 1 | MultinomialNB | 0.638690 | 0.666131 | 10000 | 0.647986 |
| 2 | DecisionTreeClassifier | 0.649834 | 0.664873 | 10000 | 0.655847 |
| 3 | RidgeClassifier | 0.730762 | 0.739152 | 10000 | 0.734125 |
| 4 | LinearSVC | 0.757846 | 0.758336 | 50000 | 0.758086 |
| 5 | MultinomialNB | 0.658424 | 0.680365 | 50000 | 0.666197 |
| 6 | DecisionTreeClassifier | 0.668170 | 0.685830 | 50000 | 0.674751 |
| 7 | RidgeClassifier | 0.742044 | 0.756513 | 50000 | 0.746957 |
| 8 | LinearSVC | 0.757589 | 0.757112 | 100000 | 0.757346 |
| 9 | MultinomialNB | 0.657282 | 0.681421 | 100000 | 0.665562 |
| 10 | DecisionTreeClassifier | 0.669952 | 0.685943 | 100000 | 0.676058 |
| 11 | RidgeClassifier | 0.742437 | 0.754869 | 100000 | 0.746859 |
| 12 | LinearSVC | 0.757935 | 0.756228 | 250000 | 0.757021 |
| 13 | MultinomialNB | 0.660640 | 0.683517 | 250000 | 0.668590 |
| 14 | DecisionTreeClassifier | 0.668623 | 0.685926 | 250000 | 0.675103 |
| 15 | RidgeClassifier | 0.744877 | 0.754769 | 250000 | 0.748600 |
| 16 | LinearSVC | 0.753187 | 0.750750 | 500000 | 0.751847 |
| 17 | MultinomialNB | 0.672800 | 0.687760 | 500000 | 0.678584 |
| 18 | DecisionTreeClassifier | 0.673318 | 0.690695 | 500000 | 0.679764 |
| 19 | RidgeClassifier | 0.742800 | 0.751770 | 500000 | 0.746270 |
| 20 | LinearSVC | 0.753156 | 0.750642 | 1000000 | 0.751769 |
| 21 | MultinomialNB | 0.673599 | 0.688095 | 1000000 | 0.679241 |
| 22 | DecisionTreeClassifier | 0.673084 | 0.688515 | 1000000 | 0.679001 |
| 23 | RidgeClassifier | 0.742725 | 0.751611 | 1000000 | 0.746170 |