Natural language processing and automatic classification: Word sentiment embedding on Twitter during a political debate

Sacha Le Doeuff, Vincent de Germiny

Université d'Avignon, CERI, 84140 Avignon, France
sacha.le-doeuff@alumni.univ-avignon.fr, vincent.degerminy@alumni.univ-avignon.fr

Abstract

This document describes the application developed for the Tweet sentiment analysis Innovation Project. The goal is to develop an application that can classify tweets by polarity, in the context of the debate between Emmanuel Macron and Marine Le Pen for the second turn of the 2017 french presidential election. This project serves as an introduction to problematics treated in domains like *natural language processing* and *supervised classification*. The system uses a machine learning model called *support vector machine*(SVM) and a *bag-of-words*(BOW) representation. In order to constitute the learning corpus, we made a polarity lexical and a python script that automatically attributes polarity scores on tweets based on certain keywords, hashtags, emojis, and punctuation marks. We also applied some basic pre-processing on the messages contained on the tweets like UTF-8 encoding, lowercase and stemming. At the time this article was written, the accuracy of the system on the test corpus ranked 5th with an average F1(accuracy measure) of **36%**.

Index Terms: classification, SVM, BOW, machine learning, stemming

1. Introduction

This document describes the application developed for the Tweet sentiment analysis Innovation Project, which goal was about making an application that can classify short messages by their polarity. These messages take the form of tweets, 240 characters messages from Twitter that will be classified between four classes:" *positive*", " *negative*", " *neutral*" and " *mixed*". All the tweets are written in french.

The four classes and their meaning are described below.

- Positive: a positive opinion or idea is expressed, usually a compliment for one of the candidates or the debate itself.
- Negative: a negative opinion is expressed, usually a critic, insult toward something or someone.
- Mixed : a more moderated opinion is expressed, usually when both a negative and a positive comment are present.
- Neutral: the message just relate facts without taking a side, no relevant information that could help determinate the polarity has been detected.

The initial corpus is composed of 70277 tweets, extracted from Twitter during the debate between Emmanuel Macron and Marine Le Pen for the second turn of the 2017 french presidential election. The polarity is impartial and will judge a message regardless of the concerned people.

Machine learning have proven, during recent years, to be quite effective at dealing with classification problems[1], but they are limited in the type of data they can treat as an input. Therefore, one of the major issues with natural language processing is being able to accurately describe a sentence in a sequence of numerical entities[2].

The extraction source of the messages also presents another challenge. Because the data is extracted from the internet, on a very popular and informal website, it is very unclean. Wrong spellings, incorrect punctuation placements and unexpected white-spaces, all participate in making the processing step more tedious. Such pre-processing phase is necessary to obtain significant result.

2. Methodologies

2.1. Corpus Study

At the beginning of the project, all participants were asked to each manually annotate 250 tweets. This step allowed us to take a first look on the corpus and to have a better understanding of the tasks which needed to be realized. The results were then used to create a ground truth for the test corpus which will be used during the entirety of the project to evaluate our predictions. The next step was to use a data exploration and visualisation tool to extract more potentially useful information from the corpus.

The tools used for the data exploration and visualisation tasks were *ElasticSearch* and *Kibana*, as well as some python scripts. *ElasticSearch* serves as a database and search engine that allows us to process data with *Painless* scripts to collect desired information with, for example, regex query. Then, with *Kibana*, we can create figures and charts with the collected information. We used it to extract a ranking of the most used hashtags during the debate which will be very useful for the constitution of the training corpus. You can see a top 10 of the most popular hashtag in the figure 1.

Python was chosen as the main language because it was a familiar language for us that allows an easy parsing of XML files and was well suited for neural networking and machine learning with the presence of numerous modules. We used the *lxml* and *xml* python libraries for XML files parsing, *French-Stemmer* from the library *nltk* for the stemming process and *unidecode* for encoding. The stemming and encoding process of the data will later be detailed in the following eponymous parts. The library *re* was also used to handle regex queries as well as *tdqm* which is used to display loading bars.

So, we took a look at some of the tweets and flew over some hashtags and words that has been used. We noted that during the debate, people tend to speak negatively. Even if all tweets haven't been annotated, it is not that astonishing to think that approximately 2 out of 3 tweet is negative. Because it is an anonymous social network, users tend to speak without any restrictions.

We found that hashtag, a special word using a # as the

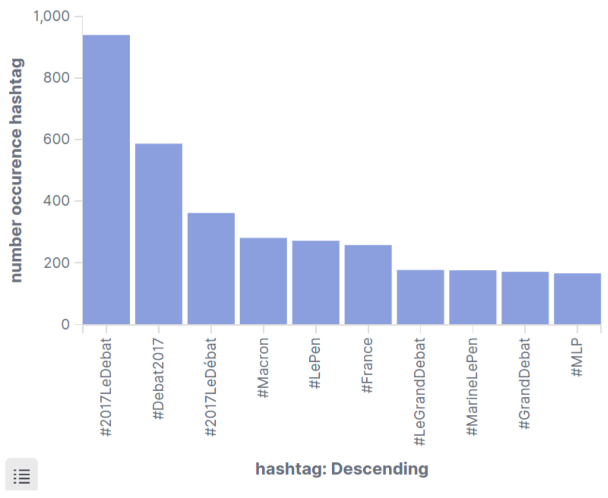


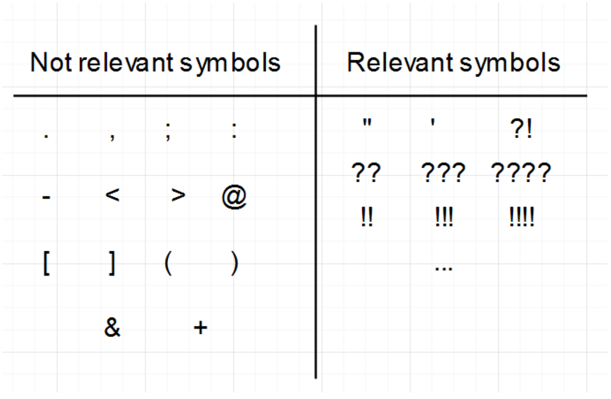
Figure 1: *Top 10 most popular hashtag*

first letter of the word, is used to put a message in the Twitter trending, and people can search message containing a specific hashtag. For example, the hashtag #2017LeDebat was the most popular during the debate, used to centralize all messages discussing about the debate. And, about what we said earlier concerning negativity, people used others hashtags to create other trends in order to criticize politicians, for example #titaniclepen or #jamaismacron.

2.2. Data Pre-processing

As explained earlier, the pre-processing phase was an important step due to the unclean and chaotic nature of the data. To create a robust representation of the messages, we had to get rid of unnecessary characters and to apply a normalization on the different words found. Not only this will reduce the side of the index and the computing time, but this will also regroup words with a sense similar and help the classifier to achieve better performances.

First, an *utf-8* encoding was applied to deal with the presence of various special characters. Then some punctuation marks and special characters considered useless were removed. Others were considered interesting and kept, more details on their impact on polarity scores will be given on ulterior parts. All of those are listed in the figure 2.



Not relevant symbols				Relevant symbols		
.	,	;	:	"	'	?!
-	<	>	@	??	???	????
[	]	(	)	!!	!!!	!!!!
	&	+			...	

Figure 2: *List of symbols*

On twitter, it is possible to mention a person with the use of the @ symbol. This information may be relevant for polarity embedding as it is possible to get the popularity of the person, but we considered this approach not reliable enough so we didn't take them into account. The URL links were also ignored.

To improve the performances, we also removed tool words which are listed in a hand-made file called *stopwords.txt*. This includes grammatical words like prepositions, pronouns or conjunctions that are not interesting for sentiment embedding.

The last step of the normalization was the stemming, which consists in replacing each words, composing the index of the corpus, by its root form (here by removing the suffix). That allows us to regroup words with similar meaning and to bypass some bad spelling. This process was also applied on the tool words file.

Now that we have applied some pre-processing, and created a clearer corpus, it is time to create a tool that can help us constitute a significant training corpus for the machine learning algorithm.

2.3. Automatic annotation

To create the learning corpus we need to annotate each tweet with a polarity. The manual approach would take too much time, because of the size of the initial corpus, so we have to elaborate a tool that automatically attributes a polarity by analyzing the content of the message. A python script has been elaborated that analyse a message and return a polarity score as output. The analysis will be based on the detection of the following types of words:

- Keywords
- Hashtags
- Punctuation marks
- Emojis (incomplete)

A dictionary for each types, containing words that have been considered positive or negative, needs to be constituted so that whenever one of them is detected we can update the polarity score. The score functions as the following: if a positive word is detected we update a positive counter, same for a negative word with a negative counter. The final score corresponds to the polarity of the bigger counter. If both counter are equals the polarity is *mixed*. If not a single word contained in the dictionary is detected, then the tweet is considered *neutral*. This method is not able to annotate every tweet, and because of this, returns an important number of *neutral*. We decided to only include tweets that have been labeled as *positive*, *negative* or *mixed* in the training corpus so that the machine learning will not train on a big number of *neutral* and be biased.

The most substantial part was the establishment of the keywords dictionary, given that even with the stemming and normalisation process there are still an colossal number of words in the corpus index (around 29 000). The keywords have been chosen following different factors, the first being their frequency in the corpus. With a python script we created a stemmed index containing words that appears at least 10 times in the corpus. Then these words have been manually classified and used to create the dictionary of which an extract is available below1. We observed that a big majority of the keywords found were negative probably due to the nature of the platform. A difficulty encountered was the presence of sarcastic tone, where a positive word was employed to give a negative opinion. Detecting such occurrence are obviously difficult but we managed to dismiss some of them by detecting the presence of a negation mark before the word like "*ne*" or "*ne pas*".

Table 1: *Excerpt of the keywords list*

Positive	Negative
excellent	insolent
meilleur	mauvais
rassuré	peur
solide	suicide
calme	regret

The hashtag dictionary have been created in a very similar way. A majority of the most used hashtags are neutral (*#2017LeGrandDebat*) but still a part of them were very effective and interesting because they tend to resume the opinion and subject of the message in one word and give obvious information concerning the polarity of the tweet (*#votezmacron,#jamaislepen*). Again a extract of the dictionary is available down below, in the table 2.

Table 2: *Excerpt of the hashtags list*

Positive	Negative
#jevotemacron	#jamaismacron
#vivelepen	#titaniclepen
#macronpresident	#macrongate
#le7maijevotemacron	#sansmoile7mai

We also tried to establish, with still the same method as before, an emoji dictionary as these characters can be very useful to detect a polarity. Unfortunately after several attempts the detection of emojis didn't seemed to work and thus we decided to give up on that aspect.

Concerning punctuation marks, most of them were considered irrelevant and were deleted but we preserved some of them that we considered interesting. The presence of numerous *'?'* or *'!'* are often used in very emotive messages and usually aggressive and negative ones. The presence of *'"'* also indicate the beginning of a citation and a tweet only composed of a citation will be neutral.

3. Tweet representation and *SVM*

As stated earlier, the machine learning algorithm chosen for this project was *SVM*. It may not be the most interesting choice in terms of performance, but its simplicity in both implementation and utilisation represents a good starting point. Plus we already had the occasion to use it in a previous practical work. We used the library *liblinear-2.30* which proposes an in-built ready to use implementation of an *SVM* machine learning algorithm. But the issue with this solution is that it only takes a very specific format of data as input, a file with an "*.svm*" extension which can only contains numerical data. Because of that we had to find a way to convert the original corpus (*XML* file) and find a representation that is suited for the library without losing too much information.

The representation chosen for the tweet was the *bag-of-words(BOW)* approach, in which a sentence is represented by a list of the words that compose it. Each word is identified by a number, corresponding to its position in the index, and by his number of occurrences in the sentence. This method is well suited for *SVM* machine learning but necessitate the creation of an index, containing every word present in the corpus, and is very basic. We lose potential interesting information like the order of the sequence of words. You can find in the figure 3 the format of each line in the input file. Each line represents a tweet's message.

```
<label> <index1>:<value1> <index2>:<value2>...
.
.
.
```

Figure 3: *Format of each line in the input file: <polarity score> <index position>:<number of occurrences>*

A single python script was used to create the index, for both the training and testing corpus as well as the bag of words representation for each of them. Because of the colossal size of the initial corpus, the creation of the index can take a while (between 15 and 30 minutes). The bag of word encoding can also take some time for the training corpus (around 10 minutes).

Now that we have suitable data for the machine learning algorithm, we can create a model and use it to predict values for the test corpus. These predictions can then be evaluated to have an insight on the performance of the model.

4. Prediction results for each version

In this section, we have detailed the results obtained on an online platform which was available for every participants of the project. This platform evaluates the predictions realized by our model for the test corpus. A python script was used in order to convert the output file given by the *SVM* into a representation acceptable by the platform, in which each line represents a tweet with its id and its polarity.

Each section will be accompanied with the detail of the list of adjustments made for this version.

The measure used to evaluate the results is the *F1-score*, also called *F1-measure*, which is a classification measure based on the *precision* (number of positive results found) and the *recall* (number of all positive present) [3].

4.1. Baseline

In this version all the tweets are annotated negatively. This only serves as baseline to compare our future implementations.

Table 3: *Baseline result*

Measure Name	result
Mean (FMeasure)	0.19792033838562%
Positive (FMeasure)	0%
Negative (FMeasure)	0.79168135354247%
Neutral (FMeasure)	0%
Mixed (FMeasure)	0%

Obviously, since the model only learn on tweets with negative polarity, we have a good results for the negative *FMeasure* but terrible ones anywhere else, which ultimately leads to an overall very bad mean result.

4.2. Version 1

In this first version, the automatic annotation was only based on hashtags. The results were not conserved as they were very bad (below the baseline). This was caused by an important number of neutrally annotated tweets in the training corpus. In this version every tweet that had no polarity score detected was still put in the training corpus, hence the important number of neutrals.

4.3. Version 2

In this version, the training corpus was only composed of tweets which has been labeled as *positive*, *negative* or *mixed*. Because of this the results were way better but we had to sacrifice the ability to detect *neutral* annotations.

The Mean *FMeasure* obtained was **0.23598783891321%**.

4.4. Version 3

In this version, we improved the automatic annotation process with detection of keywords. We also created a stemmed index and we deleted tool words.

The Mean *FMeasure* obtained was **0.26300716483244%**.

4.5. Version 4

In this version, we corrected several errors in the *bag-of-word* encoding of tweets in a format supported by the SVM. These issues were caused by the stemming process which caused duplicated entries in the index. We also improved the automatic annotation process with the detection of punctuation marks. To solve the issue of the absence of neutral tweet in the training corpus we decided to randomly add a certain number of them. Whenever a neutral is given as an output of the automatic annotation system he has 1% chance of being added to the training corpus. This allows us to have restricted number of neutral without being overflowed and biasing too much the model. This percentage is arbitrary and can possibly be tweaked to improve performances but we didn't realized enough tests to determine a optimal value.

The final and best score obtained was realized by this version and is detailed below.

Table 4: *Version 4 result*

Measure Name	**result**
Mean (FMeasure)	0.36258680358048%
Positive (FMeasure)	0.35402298850575%
Negative (FMeasure)	0.74734607218684%
Neutral (FMeasure)	0.21564482029598%
Mixed (FMeasure)	0.13333333333333%

5. Conclusions

We presented out results, and the best performance we got, using a SVM classifier and a bag-of-word model. We noticed the rise of the accuracy over the time and the evolution of the system, as well as the importance of the pre-processing step and all the decisions made for improving the automatic annotation. Our system has been ranked 5th out of 11 members participating in the group, with a f-measure of 36%.

The score can be improved by upgrading again the quantity of words or hashtags in the dictionaries,which will make the automatic annotation step more precise and accurate. The ones we made only use those who appear at least 10 times in the corpus, but there are a lot of remaining relevant words that has been used less than 10 times.

As stated earlier the bag of words representation also presents a lot of downsides and other approaches used for word embedding, like *word2vec*, could be very interesting to use in this context. These methods allow a vector space representation of words in which words with similar meanings or that share similar context are placed closed to each other. Unfortunately because of time constraints we didn't have the chance to try that solution. The SVM is also a very basic classifier and other type of neural networks like *Convolutional neural network(CNN)*[1] have proven during recent years to be quite efficient at dealing with natural language processing.

6. Acknowledgements

We would like to thank R. Dufour, M. Rouvier and V. Labatut who supported and helped us in this project, by giving us examples, explanations and by providing us and all the students the evaluation platform. Thanks to the CERI for giving us the latex template for this article.

7. References

[1] M. Rouvier and B. Favre, "Sensei-lif at semeval-2016 task 4: Polarity embedding fusion for robust sentiment analysis," *Proceedings of SemEval-2016*, p. 202–208, 2016.

[2] R. Collobert and J. Weston, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, no. 12, pp. 2493–2537, 2011.

[3] "F1 score." [Online]. Available: https://en.wikipedia.org/wiki/F1_score