# AML Lab Report

Challenge 3 - Natural Language Processing : Sentiment Analysis

Name of Student 1: Zachari Thiry
Name of Student 2: Matthieu Ramon
Name of Student 3: Benjamin Salon
Promo: 2023

AML : Advanced Machine Learning
(Spring, 2022)

EURECOM Graduate School and Research Center in Digital Science
Data Science Cursus

8th June 2022

## Abstract

Machine learning exercise done during the Algorithmic Machine Learning course at EURECOM.

Based on a collection of tweets scraped from the *Figure Eight's Data for Everyone platform*, we will try and predict the sentiment given by these small test sentences. This is a sub field of Natural language Processing. Given the nature of text documents, they can not be processed by machine learning algorithms. For this reason, we will start by transforming the data set into numerical features, first using the given "Bag of words" techniques and then moving on to the BERT Google's state of the art NLP algorithm reaching a final F1-beta score of 0.84.

## Contents

# 1 Data Analysis

## 1.1 Introduction to the data set

The given data set is a collection of roughly 27 500 text samples among which 2 500 ( one 10th ) for testing. Each text has a "*selected_text*" feature which consists of an extract from the initial text, the sentiment of which is a little clearer to deduce.
**Example :** Hello, yourself. Enjoy London. Watch out for the Hackneys. They‘re mental.
Where only the underlined part is kept for training.

## 1.2 Analysis and first thoughts

Let's start by getting a little information on the dataset.

- **Data type :** Machine learning algorithms are made to work on numerical features. However, our data set is comprised on text only. As such, we will need to convert the data into numerically exploitable features. Many algorithms exist for this purpose. They will be detailed into the Data Pre-Processing section (2.).

- **Task :** The given task is sentimental classification of text within three classes : Positive (1), Neutral (0), Negative (-1). Such tasks are very sensible to skew in the data and equal probability of classes. Let's take a look at what we have :

| **Class** | **Proportion** |
|---|---|
| Positive | 31% |
| Neutral | 41% |
| Negative | 28% |

Figure 1: Class distribution in the data set

These results tell us that although the distribution isn't perfectly uniform, there are enough of each samples to train the model correctly. Implementing distribution equalizing techniques (such as over or sub sampling) might improve the results but aren't necessary to begin with.

# 2 Data Pre-Processing

## 2.1 Raw execution : feature selection

One of the easiest transformations to do on a textual data set is called *"Bag of Words"* and consists in the following : A dictionary of the words found in the training data will be created and form the features. Then, a one hot encoding is applied to each row. Let's see an example :
Given the following two sentences :

- "I like birds."
- "Birds like to eat worms."

- "Worms like to eat birds."

The output transformed data set is the following matrix :

| Row N° | i | like | birds | to | eat | worms |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 1 | 1 | 1 | 1 | 1 |

Figure 2: Example of Bag of Words

We will note the following remarks :

- The order of the words isn't considered with this algorithm : this means that two opposed sentences with completely opposed meanings could be interpreted the same way by the algorithm ( see rows 1 and 2)
- The bigger the dataset, the higher the dimensionality of the input and the more sparse the matrix. This is not scalable.

As such, we used the bag of words technique on the "selected_text" feature : it is more concise, contains less words and its interpretation is "humanly" less ambiguous : the results show that this feature is more efficient on simple model such as Naive Bayes Classifier.

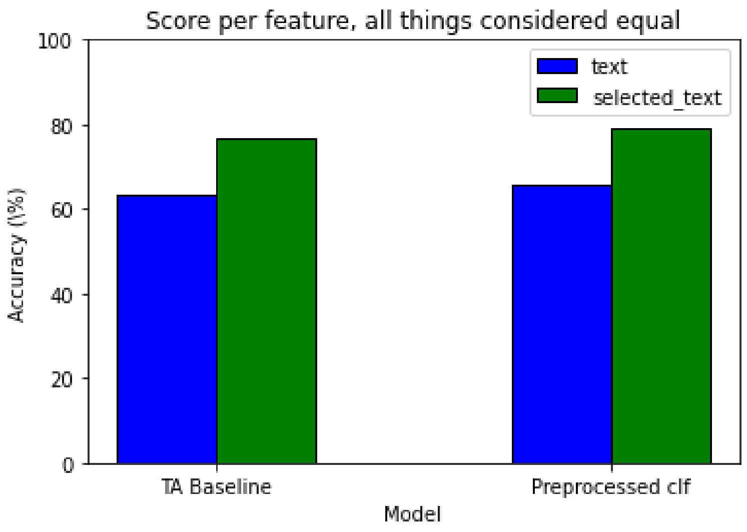

Figure 3: Accuracy of models given different inputs

## 2.2 Transformation of the data set

Given the nature of the bag of words technique, the resulting input data is highly dimensional : this could be a problem. For instance, raw implementation of the bag of words leads to a input that has more features than data points.
A quick look on the web led us to use the NLTK library. It is widely used, provides a good documentation, and many implementations of it are freely

available. The goal is ambivalent : reduce the dimension of the input and group together words that have the same semantic meaning. Here is what we did :

- Applied a pre-processing filter that removed punctuation, transformed all the strings into lower cases and removed numbers,
- Filtered the most common stop_words using NLTK English stop words assuming that all the tweets were in English ( A non exhaustive manual search was performed and revealed that the tweets were only written in the English language)
- Tokenised and lemmatised the sentences so that words of the same root will be grouped together.

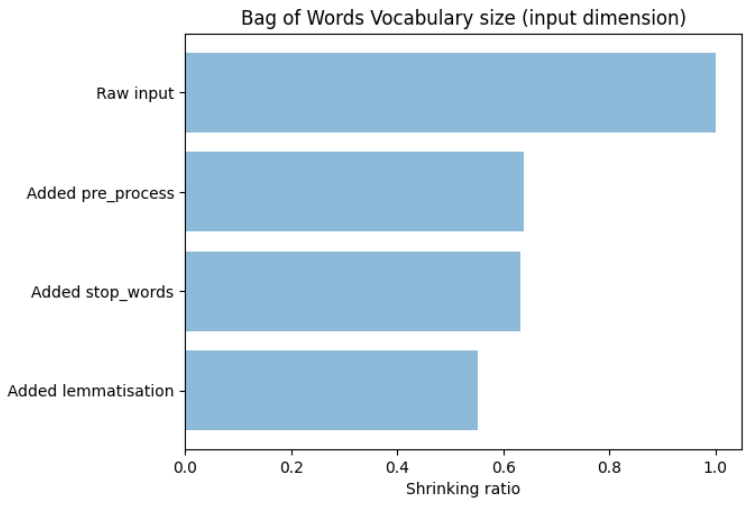

Figure 4: Effects of the pre-process on input dimension

## 3 Model Selection and results

### 3.1 Two approaches: Naive Bayes Classifier and BERT

We decided for this challenge to use, as a first approach, the given baseline of a Naive Bayes Classifier to then have the comparison with an implementation of the state of the art algorithm BERT algorithm. The first algorithm also allowed us to get quick iterations to assess the efficiency of the different data preprocessing methods, thus, the results obtained thanks to the Naive Bayes Classifier are already showed and detailed in the preprocessing section and so we will focus in this section on the BERT model implementation.

### 3.2 BERT: Bidirectional Encoder Representations from Transformers

BERT is a state of the art NLP algorithm developed by Google Research in 2018. It is declined in two different versions the BASE model and the LARGE model, the LARGE model being the most advanced one with 340M parameters where the BASE has 110M. For this challenge we decided to use the BASE model as it is easier to train and less computationally demanding. The big advantage of this model is the benefit of transfer learning it provides. The models can be found already pretrained with a huge data set consisting of texts from Wikipedia and Google Book's corpus. We only then have to finer tune our models to adapt them to the particular task we want to solve.

### 3.3 Results

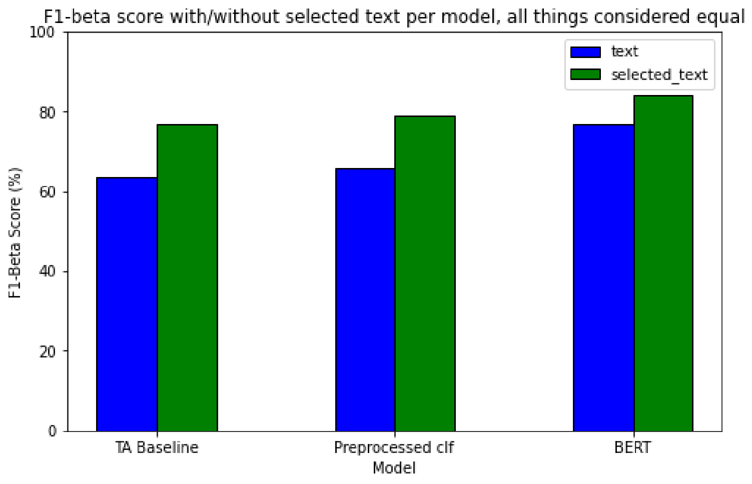

Figure 5: Comparison of F1-beta scores

Thanks to the implementation of the BERT algorithm we manage to raise our submission F1 score from the 0.76 we reach with the Naive Bayes Classifier algorithm with preprocessed data to the 0.84 which will be our best submission for this challenge.

We can see here the clear gain we have thanks to the selected_text preprocessing. We gain 0.07 on the submission F1 score going from 0.77 to 0.84. We can also notice the power of the BERT model which gives, without preprocessing, a result equal to the result of the Bayes Classifier with preprocessing.

## 4 Limits reached and self analysis

- **BERT implementation** Implementing a pretrained algorithm like the BERT algorithm was not so easy due to many versions incompatibility issues encountered when trying to use the tensorflow library. We finally managed to implement it pytorch based. We learned that a public model like this one can be exploited via different libraries.
- **Organisation** This challenge has been a very significant improvement in our overall organisation scheme. We managed to well distribute the tasks among us and to be much more efficient in the implementation than the previous challenge.