Machine Learning Project 2

Cécile Chavane, Jehan de Bryas, Antoine Goupil de Bouillé
Projects EPFL, Machine Learning Course, Fall 2022

***Abstract*—This report develops the preprocessing and modeling to binary classify Tweets based on the feeling their convey. This report will explore the different methodologies and results that we achieved.**

I. Introduction

This second project aims at making supervised binary classification of tweets according to the emotion given by this phrase. These tweets initially contained positive smiley ":)" or négative smiley ":(". These emojis has been removed from the tweets and the tweets has been labeled according to the emoji removed.

In order to classify this text data, we need to develop a methodology to transform these tweets into matrices, then build classifiers trained on the data, to finally be able to make predictions.

Our understanding of the transversal objectives of the project seems to be structured around several key points: understanding the functioning of the basic machine learning mechanisms: preprocessing of data, visualizations, variables, selection of the best drivers, elaboration of models, selection and evaluation of models and parameters, training on the whole data set and finally predictions on the test data.

We believe that this project has many real world applications. Social medias aim to reduce anger, racism or other forms of discrimination, and therefore use algorithms to detect such form of speech. This kind of tool can also allow to detect fake news information for example.

Our report will try to clarify the functioning of our classification model, by evoking particularly the decisions of preprocessing and methods of selection of the models, allowing to reach the results that we had.

II. Vizualisation of the data

A. Vizualisation of vocabulary between positive and negative tweets

The first graph represents the most common words in the positive Tweets whereas the second one gives the most common words in negative Tweets. Also

B. Vizualisation of the length of tweets (words #)

Fig. 3. is a histogram of the length of tweets, according to their labels.

III. Preprocessing the dataset

Preprocessing of the dataset is mendatory in Natural Language Processing (NLP) in order for the machine to be able to understand text. The main goal is to apply a mapping between words in the Dictionnary (the list of all words in the training dataset) and numbers, to be able to transform sentances into Matrixes, in the vectorial space of our Dictionnary.

Fig. 1. Most common in positive tweets

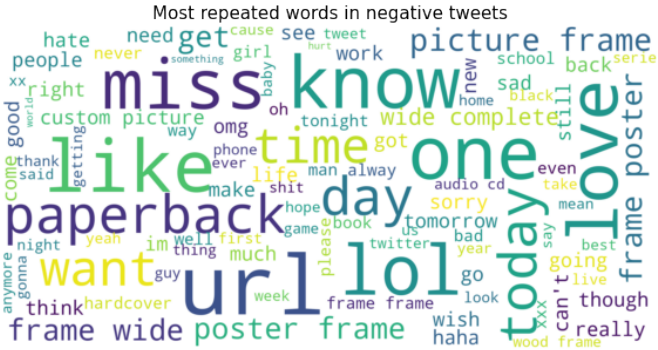

Fig. 2. Most common words in negative Tweets

Other transformation of the data can be very usefull to improve our machine learning predictions. It is mendatory to apply the same transformations to the whole data set (training and testing data).

In this part of the report, we will explain the transformation that we used and the results of each of theses preprocessing methods.

A. Cleaning the Data Set

The process of cleaning the data is mendatory and will be common to all models and preprocessing methodologies.

1) Labelling the data: When the Tweet corresponds to a positive emotion, we assign the label 1. When the Tweet corresponds to a negative emotion, we assign the label -1.

2) Text cleaning: We have had to clean the text data in Tweets in order to reduce the number of characters non-usefull for the understand on the sentences. Therefore, we applied text

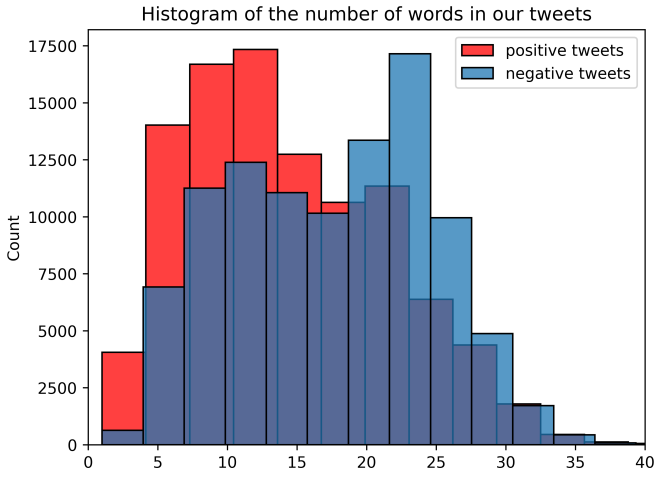

Fig. 3. Histogram of the number of words of labeled tweets

modifications to delete all special characters (e.g. ”/”, ”?”, ”!”, etc.), transform all words to low characters, delete single letter words. We also delete url links and numbers, because they do not convey usefull information for sentimental analysis.

3) Removing duplicates: Moreover, we have chosen to delete the duplicate Tweets (most of them coming from Re-Tweets of famous people’s Tweets). Indeed, theses repeated values would give extra important to some words compared to others and therefore reduce the prediction ability.

B. StopWords

Stopwords are the words in a stop list which are filtered out during preprocessing. There is no single universal list of stop words, and the choice of the list depends on the application. The question of keeping or removing stopwords is very frequent in NLP problems. More importantly, the choice of the list will affect the prediction power of the model. Deleting ”non usefull” words from the model will reduce dimension, but also could the model not understand underlying relationships correlated to these words.

1) Twitter stopwords: Using wordcloud vizualisation of most frequent words in Tweets, we can have a first insight of usefull and unusefull words. Before preprocessing, we can observe that TWitter specific words appear very frequently, for example ”user” or ”rt”, which means ”Re-Tweet”, the action of sharing back an existing Tweet. We have chosen to delete these words that don’t convey information about the feeling of the Tweet.

2) Common English StopWords: Famous NLP Python libraries such as NLTK, provide users with StopWords lists, containing a selection of ”non-significant words”. It appears that some words in the list could be very significant in order to classify positive and negative Tweets. Indeed, all the negative forms of verbs (”hasn’t” vs. ”has”) give usefull information of the polarity of a phrase. Moreover, preprositions and adverbs (”not”, ”nor”, ”any”) are also usefull in this situation. We therefore have modified the existing NLTK English StopWords list.

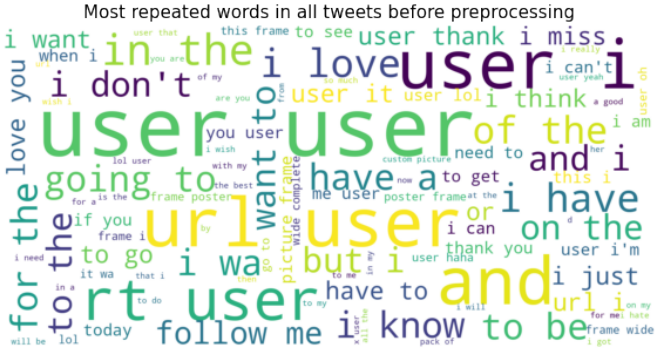

Fig. 4. Most common words before preprocessing

Fig. 5. Most common words before preprocessing

C. Text Normalization

The main goal of the text normalization is to keep the vocabulary small, which can help to improve the accuracy of our predictions, and to improve the time of training.

1) Lemmatization: Lemmatization considers the morphological analysis of the words and converts the word with meaningful way into the radical noun. Lemmatization has the advantage of keeping the meaning of words. We have used the Lemmatizer provided by NLTK library: WordNetLemmatizer.

2) Stemming: Stemming usually refers to a process of chopping off the last few characters.

Stemming can have bad effects for prediction. Overstemming comes from when too much of a word is cut off, and therefore is not accurate. Understemming comes when the word is not cut enough. We have tested two different stemmer algorithms. Porter Stemmer removes the common endings to words. Snowball stemmer is almost universally accepted as better than the Porter stemmer.

D. From text to trainable arrays: vectorisation

In order to make text understandable for the machine, we need to create a mapping between the words in the vocabulary list we chose, composed of all unique words in training data

set reduced from the stopwords list. The simple solution for vectorization is a co-occurence matrix. However, it is usefull for better predictions to create a mapping based on the meaning or occurences of words. Thereore, two words with similar meaning or usage will be close. We have used GloVe Embedding method and RoBerta Transformer to achieve this using pre-trained models.

1) TFIDF Matrix: A TFIDF matrix is a numerical representation of a collection of documents that reflects the importance of each word within and across the documents. It is calculated by multiplying the term frequency (number of times a word appears in a document) by the inverse document frequency (logarithmically scaled inverse fraction of documents containing the word). The resulting TFIDF value reflects both the importance of the word within the document and its importance across the collection of documents. The matrix has a row for each document and a column for each word, with the value in each cell representing the TFIDF value for that word in that document. This way of representing trainable data has been used to train our sklearn models (i.e. logistic regression, descision tree and the multi-layer perceptron classifier)

2) N-grams vectorization process: N-grams vectorization process is based on a simple co-occurence matrix. However, this methodology adds N-grams, continuous sequences of words or symbols or tokens in a document. N-grams vectorization process is only used with the TFIDF matrix. Hence, we used it all our models using TFIDF matrices, and it improved our accuracy. This can help understand complex gramatical forms, such has "has not", that conveys a negative idea. This meaning is not understandable without using sequences of 2-3 words.

3) Padding sequence: Padding is a technique used in natural language processing (NLP) to handle input data of varying lengths. It involves adding extra elements (such as zeros or special tokens) to the input data so that all the input sequences have the same length. Padding is often used when working with neural networks, as they require fixed-sized inputs. The added elements are typically placed at the beginning or end of the input sequence, depending on the padding strategy being used. Padding is an important step in preprocessing data for NLP tasks, as it allows the model to more easily learn patterns and relationships in the data. This technique allows neural network to capture the context of each tweet whereas the above TFIDF technique can't. Padding sequence representation of data has been used to train our bi-LSTM model.

4) BERT and BERT-variant transformers: BERT, or Bidirectional Encoder Representations from Transformers, is a natural language processing model that produces state of the art results in a large range of NLP tasks. BERT uses a specific tokenization method that takes into account the structure of the input language and is designed to preserve the meaning of the text as much as possible. The resulting tokens are then fed into the BERT model for further processing. We will use this BERT Tokenization to fine tune Transformers from Hugging face.

5) Other methods: Some token-izing functions are integrated into packages in the model used. These methods will not be described in this part, but use all the above methods.

E. Principal Component Analysis

The PCA is used to reduce the number of drivers by keeping only the principal components directions. The dataset indeed contains thousands of different words, therefore implicating high dimension. We have tries several values of drivers. The PCA methodology reduces the accuracy but accelerates the process of training.

F. Comparing preprocessing methods

1) Approch: We compared the impact of each preprocessing method by comparing two logistic regression models, one with data preprocessed with the given method, the other without. If the method allows a better accuracy score, we keep this method and apply it to the following tests. We first tested the data cleaning, then the stopwords removal, the normalization methods and finally the vectorization. This methodology does not take into account all the possible combinations, but ensures to measure the impact of each method.

Method	With	Without	Choice
Special characters	79,3%	79,0%	With
English stopwords	78,3%	79,0%	Without
Lemmatization	78,0%	78,4%	Without
Porter Stemmer	78,4%	78,4%	With
Snowball Stemmer	78,0%	78,4%	Without
2-Grams	79,5%	78,0%	With
3-Grams	79,2%	78,0%	Without
PCA 10 000 drivers	79,5%	70,6%	Without

TABLE I
COMPARING PREPROCESSING METHODOLOGIES

2) Results: We finally chose to keep the methods with the label "With".

IV. MODELS AND METHOD

All the results of the methods below are listed on table III.

A. Logistic regression

Logistic regression is the first model used for the classification of observations. In statistics, logistic regression is a binomial regression model. We have used gradient descent to find the best parameters to fit the training data.

B. Descision tree

We thought that random tree could be a good classifier for our tasks. After some optimization of the paramters, and taking into account the training time that augment very fast with the depth of the tree, we choosed a depth of 100.

C. Simple Multi-layer Perceptron classifier

A multi-layer perceptron (MLP) classifier is a type of artificial neural network that is used for supervised learning tasks, such as classification. It is called a multi-layer perceptron because it consists of multiple layers of interconnected "neurons," which are mathematical units that process input and generate output. We decided to implement it with two hidden layers of sizes 30 and 20. The activation function used is relu.

D. Soft Vector Machine (SVM)

SVM is a supervised learning method that looks at data and sorts it into one of two categories. An SVM outputs a map of the sorted data with the margins between the two as far apart as possible.

E. Bi-LSTM model

This time, using padding sequences and modeling using keras library, we trained a neural network for which the model is summarised here :

Layer (type)	Output shape	Activation	params
text (InputLayer)	[(None, None)]	None	0
embedding$_1(Embedding)$	(None, None, 128)	None	1280000
bidirectional$_1(BidirectionalLSTM)$	(None, 200)	tanh	183200
dropout$_1(Dropout)$	(None, 200)	None	0
dense$_2(Dense)$	(None, 30)	relu	6030
dense$_3(Dense)$	(None, 1)	sigmoïd	31

TABLE II
BI-LSTM MODEL SUMMARY

F. Fine tune transformers models from HuggingFace

Fine-tuning a transformer is a process of adjusting the pre-trained model on a new dataset for a specific task. Transformers are a type of deep learning model that have achieved state-of-the-art results on a wide range of natural language processing tasks, such as language translation, language generation, and text classification. Fine-tuning a transformer involves using the pre-trained model as a starting point and adjusting the model's parameters on the new dataset in order to optimize it for the specific task at hand, which is here for our twitter dataset. Transformers library from hugging face provide objects and function scpecially designed to fine tune transformers models from a large range of BERT variant models. Here we have decided to fine tune these pre-trained models from Hugging face :

- 'bert-base-uncased'
- 'roberta-base',
- 'ProsusAI/finbert',
- 'distilbert-base-uncased-finetuned-sst-2-english',
- 'finiteautomata/bertweet-base-sentiment-analysis'.

These pre-trained models, after fine tuning, gave various performances reported on table III.

G. Selection of hyper-parameters

We have used a K-fold cross validation methodology to train and test our models and be able to find the best hyper-parameters. The K-fold cross validation process gives the average of all folds. We have tested different values of k, but 5 gives us good results to compare the models.

V. Results

Here is the table of the results to compare different models test accuracy. Finally , we decided that the finiteautomata/bertweet-base-sentiment-analysis transformers model was the best for our task and submitted it to AICrowd. It gaves us **86.7%** score for Submission #207476, by Jdbrs account.

Model	Test accuracy	F1 Score
Logistic regression	80.8%	xx%
random trees	76.6%	xx%
MLP classifier	81.1%	xx%
Bi-LSTM model	83.4%	xx%
bert-base-uncased	83.0%	xx%
roberta-base	83.4%	xx%
ProsusAI/finbert	78.6%	xx%
distilbert-base-uncased-finetuned-sst-2-english	77.4%	xx%
finiteautomata/bertweet-base-sentiment-analysis	**88.0%**	xx%

TABLE III
COMPARISON OF MODELS

VI. Summary

A. Recommandations

B. Possible improvements