Text Sentiment Classification

Mariem Belhaj Ali, Nourchene Ben Romdhane, Mouadh Hamdi
Team Name : *who am I*
Department of Data Science, EPFL Lausanne, Switzerland

***Abstract*—Focusing on the Twitter social platform, this project aims at predicting if a tweet message used to contain a positive ☺ or a negative ☹ smiley. This is also known as sentiment analysis. In this paper, we will explain our choice of solutions, including exploratory data analysis, text preprocessing, and feature extraction. We will use and combine machine learning methods, specifically supervised classification algorithms, to generate our predictions and get the highest correspondence with the reality.**

I. Introduction

Sentiment Analysis is a field within Natural Language Processing (NLP). It tries to identify and extract opinions within text, in addition to extracting attributes of the expression, such as polarity (the speaker's opinion is negative, neutral or positive) or entity (the thing or person that is being talked about). Today, sentiment analysis systems allows companies to make sense of this sea of unstructured text and get key insights on the users' thoughts.

Twitter, being the 12th[1] most visited website in the world, with more than 326 million monthly active users, is one of the most popular social networks. In this project, we will analyze 140 characters tweets, and decide whether they contained a positive or negative smiley. To do so, we will use different preprocessing techniques in order to treat text, emojis, special characters, and hashtags contained in tweets. We will then study different text representations, and finally analyze various classification algorithms in order to generate the best results.

II. Data Description

We were given 2.5 million tweets that used to contain either a positive or a negative smiley. This data set will be used for training, and we were also given a test data set consisting of 10'000 tweets, which will be used to generate the predictions, tested on corwdAI. When inspecting the data, we can see that the majority of the tweets consist of non formal words (slang words), special characters, hashtags, urls, and these need to be delt with, which will be explained in the following part.

III. Data Preprocessing

Here, we will be explaining the different steps of preprocessing the tweets. As said above, they consist of raw text and special characters that need to be dealt with. The methods used can be found below :

Removing urls and users: tweets contained urls replaced by <url> in the raw data set, so we remove them. They do not play any role in sentiment analysis. The same thing goes for <user>.

Removing hashtags: Since tweets contain # followed by a word or a combination of words, which are called hashtags, we remove the # and leave the word or split it if it was a combination of words. To split the concatenated words we used the library wordninja. [2]

Emphasizing the sentiment: Some words have a clear polarity, so using a set of positive and negative words, we added the word 'positive' whenever one of the words in the tweets was in the positive set of words, and negative if it was in the negative one.

Emojis Translation: Since tweets can contain many emojis, we chose to analyse their meanings to get more insight on the sentiment shared by the user. We implemented a method for emojis transaltion, in order to classify all possible emojis into 3 different sets : smiley faces, sad faces, and neutral faces. We also treated the case of heart emojis. As the words in the tweet are gradually treated by the method, if an emoji is found, we transform it into a word according to the set it was in (happy, sad, love, neutral).

Punctuation Treatment: We clearly seperate the punctuation from the words with space, and then delete them. This can be quite controversial since punctuation can sometimes be used to send a clear sentiment or opinion, but for our model we chose do disregard it.

Small words removal: We only leave words that have a length bigger than 1 (i.e. that are not only letters). We indeed do not consider that a letter can reflect a sentiment.

Slang words treatment: Some words, like *brb* or *4u* for example are commonly used as short for *be right back* or *for you*. These shortned words are commonly known as slang words, and we made the choice of using a slang words dictionnary to return them to their original meaning.

Apostrophe treatment: Words like *it's*, *i'd* which represent *it is* or *i would* for example, are returned to their original form.

Repetition handling: There can be words with many repetitions of the same letter, like *llooooovee*, which clearly mean *love*. We chose to remove the unnecessary redundancy.

Spelling mistakes: Correction of common spelling mistakes, like *achivements* instead of *achievements*, *actin* instead of *acting*, etc...

Numbers removal: We chose to remove numbers, since after all the common words treatment, if there are some left, they

[1] https://www.alexa.com/siteinfo/twitter.com

[2] https://pypi.org/project/wordninja

do not act as strong sentiment polarizers in our opinion.
Stopwords : We chose to remove the stopwords (the, a, ...) from the text since they only help make sense of a phrase and they do not actually play a role in its polarization.
Lemmatization and Stemming: So words can have the same weight if they are just used in different tenses, we chose to lemmatize and stem our text.

IV. Vector representation of tweets :

In this section, we will present the techniques we used to transform the tweets into several numeric-vector representations that depicts significant characteristics of the text.

N-gram: The n-gram model is integrated in most text classification tasks. This model can take into account sequences of words in contrast to just using singular words.
TF-IDF: The word's importance in a tweet can be captured by the Term Frequency-Inverse Document Frequency representation, which computes the weight of a word according to the number of times it appears in the tweet and then all the tweets.
Word embeddings: In this representation, similar words or words that appear in the same situation are mapped to high dimensional vectors that are close to each other in the space. We will mainly talk about pre-trained GloVe embeddings, and the generation of other ones using GloVe. GloVe is an unsupervised learning algorithm for obtaining vector representations for words.

1) *Pre-Trained GloVe embeddings:* We can use pre-trained Twitter word vectors[3]. These vectors were trained using 2 billion tweets and a vocabulary of 1.2 million words. We chose to work with vectors of dimension 200.

2) *GloVe embeddings:* We used the glove package in python to create our own word vectors embeddings from co-occurrence matrices.

From word embeddings to tweet embeddings:
We can construct vectors for the tweets by multiplying the word vector obtained with the gloVe embedding by the tf-idf weight. We then aggregate by performing the mean of word embeddings of all the words in the tweet. Those vectors will be essential later on for the models using the TF-IDF representation.

For all the representations above, we estimated the maximum number of words a tweet can consist of (nb_word), we changed the dimension of all the vectors to be equal to this maximum number, and we padded with zeros the vectors that had less words.

V. Models

In this section, we will describe all the employed models according to the , which are 13 in total :

[3]https://nlp.stanford.edu/projects/glove/

A. Using n-grams model:

For high n values, we will need more resources and computation power to train our model. We only used unigram and bigram representations.

B. Using the TF-IDF representation:

For the TF-IDF representation we have two forms, the first one is the TF-IDF vectors generated by TfidfVectorizer. The second one is the tranformations from word embeddings to tweet embeddings.

For those representations we used a pipeline from sklearn.

Pipelines: provides a usefull layer of abstraction for building complex estimators or classification models. Pipelines also have interesting features such as fit, transform and predict.

The folowing Models were passed to the pipeline:

- **LinearSVC**
- **LogisticRegression**
- **PassiveAggressiveClassifier**
- **VotingClassifier**, which uses majority rule voting from the previous models.

C. Using the pretrained GloVe representation:

In this part, we used the pretrained GloVe embeddings since they gave us a 0.2 increase in accuracy approximately.

Neural Networks:

Neural Networks or connectionist systems are computing systems, based on a collection of connected units or nodes called artificial neurons. It is constituted of an input and an output layer, as well as multiple hidden layers :

1) *Embedding Layer:* The Embedding layer is initialized with at least 3 parameters : the maximum size of the vocabulary (input_dim =1 + nb_word), the the size of the vector space in which words will be embedded (output_dim), and the length of the input sequence (input_length). Given these parameters, it will learn an embedding for all of the words in the training dataset.
2) *Dense Layer:* A dense layer is a layer where each unit or neuron is connected to each neuron in the next layer. Each neuron recieves input from all the neurons in the previous layer. The layer has a weight matrix giving the importance of the input, a bias vector, and the activation functions of the previous layer. The activation function of a node defines the output of that node, given the inputs. It maps the resulting values into the desired range (-1 to 1 in our case).
3) *Dropout Layer:* Dropout is a a technique used to overcome overfitting . Dropout takes in a float between 0 and 1, which is the fraction of the neurons to drop.

Convolutional Neural Networks:

A CNN is a special case of Neural Networks, most commonly applied to analyzing visual imagery. A convolutional layer is added in addition to the previously described ones in the NN section. "Originally invented for computer vision, CNN models have subsequently been shown to be effective for NLP and have achieved excellent results"[4]. We hence chose to use them on our pre-processed tweets and see if they give us good predictions.
According to the glove representation, we hence have can have a matrix representation with dimension nb_word x D for each tweet, where D = 200 is the dimension of the GloVe word embeddings. Here's an overview of the added layers for the CNN models :

1) *Convolutional Layer:* The convolutional layer is the core building block of CNN models, it is used for feature extraction. It takes as parameters the number of filters, the size of the filter and the activation function. Numerous convolutions are performed on the input, where each one uses a different filter. This creates different feature maps, which are then combined to output the final result. We used the Convolution1D layer.
2) *Pooling Layer:* The pooling layer is used to reduce the spatial dimensions of a convolution neural network model. We used the MaxPooling1D layer.
3) *Flatten Layer:* Flattening is the process of converting the output of the convolutional step (multi-dimensional arrays) into a single 1D feature vector, to be used by the dense layer of the NN for the final classification.

Recurrent Neural Networks:

A recurrent neural network (RNN)[5] is an extension of a conventional feedforward neural network, which is able to handle a variable-length sequence input. The RNN handles the variable-length sequence by having a recurrent hidden state whose activation at each time is dependent on that of the previous time. It's used to memorize information for a long time period using the hidden layers and the important number of neurons.

1) Long Short Term Networks (LSTM): Helps preserving the error that can be backpropagated through time and layers. LSTMs contain information outside the normal flow of the recurrent network in a gated cell. Information can be stored in, written to, or read from a cell, much like data in a computers memory. We used for LSTM models different hidden layers, sizes of memory cells(50, 100) and dropout rates(0.2, 0.3).
2) Gated Recurrent Unit (GRU): We decided to test GRU even though it can be considered as a variation of the LSTM. GRUs solve the vanishing gradient problem of a standard RNN using the update and reset gate techniques.

[4]https://www.aclweb.org/anthology/D14-1181
[5]https://arxiv.org/pdf/1412.3555v1.pdf

VI. Results:

Choice of the preprosessing technique:

We tried different combination of data cleaning that yielded different results which will be described in the tables below. We will only give you the result for the best and the second best approaches.

1) Best approach: The best data cleaning was to process hashtags, apostrophes, correct spelling, correct slang words, remove repetition and emphasizing. See Table II
2) Second best: process hashtags, apostrophes, correct spelling, correct slang words, emphasizing, emojis, remove repetition and clean ponctuation. See Table III

Models for tf-idf representations:
In the figures below, you can see the accuracy scores obtained when using the tf-idf representation on a pipeline of models:

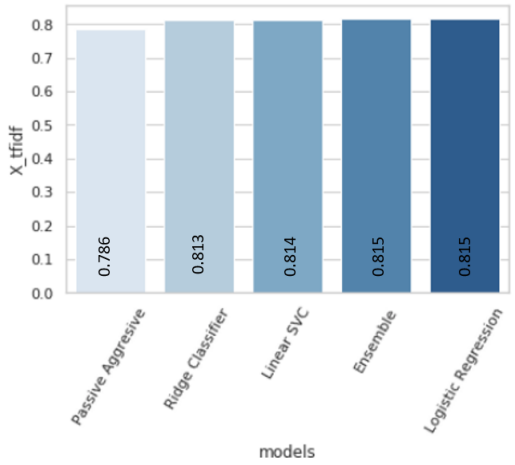

Fig. 1. Accuracy with TF-IDF

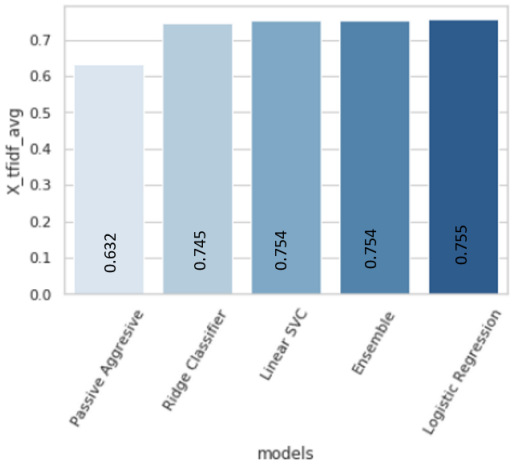

Fig. 2. Accuracy with tweet embeddings

Choice of hyper-parameters:

We tuned the hyper-parameters in each model in order to get optimal results. We hence have the following values for all the hyper-parameters of all the models:

TABLE I
HYPER-PARAMETERS TUNED VALUES

Hyper-Parameter	Possible Values	Best Value
Number of filters	32,64,128	32
Batch size	32,128,256	128
Number of epochs	1,2,3,4,5	2
Droput rates	0.2,0.3,0.5	0.2
Filter length	2,3,5	3
Activation function	binary step, relu, sigmoid	relu, sigmoid
Optimizer	sgd, adam, adadelta	adam

The table below shows the best values associated to each model according to the best approach :

TABLE II
BEST APPROACH RESULTS

Best Models	Accuracy	Validation Accuracy
CNN + pretrained glove	88,00%	87,01%
LSTM + pretrained glove	88,32%	87,38%
NN without glove	83,42%	82,80%
CNN without glove	88,13%	86,89%
LSTM without glove	88,15%	87,12%
GRU + pretrained glove	88,25%	87,29%
CNN + GRU without glove	88,32%	87,29%

The table below shows the best values associated to each model according to the second best preprocessing technique:

TABLE III
SECOND BEST RESULTS

Best Models	Accuracy	Validation Accuracy
CNN + pretrained glove	88,12%	86,98%
LSTM + pretrained glove	88,39%	87,33%
NN without glove	83,45%	82,79%
CNN without glove	88,20%	87,03%
LSTM without glove	85,45%	86,73%
GRU + pretrained glove	88,30%	87,26%
CNN + GRU without glove	86,32%	87,01%

Ensembling results from different models

After testing the previous models, we tried to use different combinations of our models to have better results. For that purpose we used XGBoost : Extreme Gradient Boosting, which is built on the principles of ensemble modeling and is an improved version of the Gradient Boosted Machine algorithm.

- Best accuracy for the first table gave us 87,0% in CrowdAi
- Best accuracy for the second table gave us 86,5% in CrowdAi

VII. CONCLUSION:

In this report, we explained different preprocessing, text vectorization and representation, and model options in order to answer the tweet sentiment classification problem. However, sentiment analysis is not 100% reliable, and is facing challenges that machine learning algorithms still can't outcome to this day. For example, analyzing a sentiment without knowing the context is quite difficult, just like analyzing sarcastic or ironic content. We can conclude that sentiment analysis is a tremendously difficult task even for humans, but it is rewarding regardless when used correctly, giving very satisfying results (generally above 80%).

REFERENCES

[1] Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, *Universit de Montral CIFAR Senior Fellow*.
[2] https://www.alexa.com/siteinfo/twitter.com
[3] https://www.aclweb.org/anthology/D14-1181
[4] https://arxiv.org/pdf/1412.3555v1.pdf
[5] https://nlp.stanford.edu/projects/glove/
[6] https://pypi.org/project/wordninja
[7] https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be
[8] https://skymind.ai/wiki/lstm
[9] https://arxiv.org/pdf/1603.02754v1.pdf