OpenClassroom project 7: Air Paradis - Detecting the Bad Buzz using Deep Learning techniques

Introduction

Sentiment analysis is highly used in NLP (natural language processing) and machine learning in order to predict if a text contains negative or positive emotions. Sentiment analysis can also be found as "opinion mining" or "emotion artificial intelligence". Sentiment Analysis can help us understand the global satisfaction about a brand, product. It is generally difficult to make this task manually because of big data that can come from everywhere like (social media posts, customer or employes reviews). The sentiments data obtained from theese sources can be used to in future to determine some bussiness key decision.

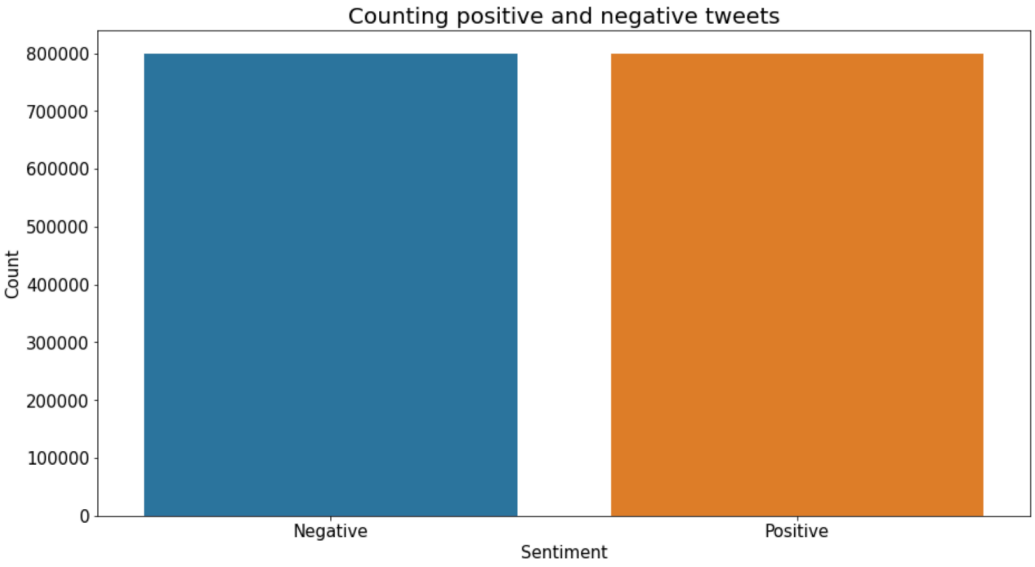
This article describes 3 different techniques to show how we predict a sentiment associated to a tweet. The first approach is about ***classical approach*** where we can use models like Logistic Regression, Random Forrest Classifier, LGBM Classifier and XGBClassifier, etc. The second approach is the ***advanced custom models with neuronal networks*** and the third approach is the ***advanced BERT*** models.

Data

Data description

We use OpenSource data that is available here. The 1st and the 5th feature is selected that contains tweets and the sentiment itselt (positive=4 and negative=0).

```
data = pd.read_csv("training.1600000.processed.noemoticon.csv",
       usecols = [0, 5], names=['Target', 'Text'], header=None,
       encoding='ISO-8859-1')
```



Data cleaning

Preparing and cleaning the data is one of the most important step in data analysis. This plays a great role in our future work so a great attention is given for this step.

Label processing: First, note the positive tweets are labeled with 4 and negative tweets are labeled with 0. We change it so that negative tweets coresponds to 0 and positive tweets to 1.

Tweet processing: We are thinking about *what are the words that mostly describes the sentiment?* and the oposite *what the words that are not helping us to give the sentiments prediction?* in order to avoid them in our data. Therefore we make some text preprocessing using NLP with *spacy* package. This step contains:

- lematization (giving a neutral form for the words)
- make lowcases,
- delete stopwords, punctuations, spaces, emails, urls, etc.

Also we will correct some words from the text.

This step creates text dataset that needs to be next vectorized and modeled.

Data vectorization

Many types of vectorization text data exists: TfidfVectorizer, Doc2Vec, Word2Vec embedding, Glove embedding, etc. The data vectorization is used in used for the first two types of modeling that is classical and advanced custom models.

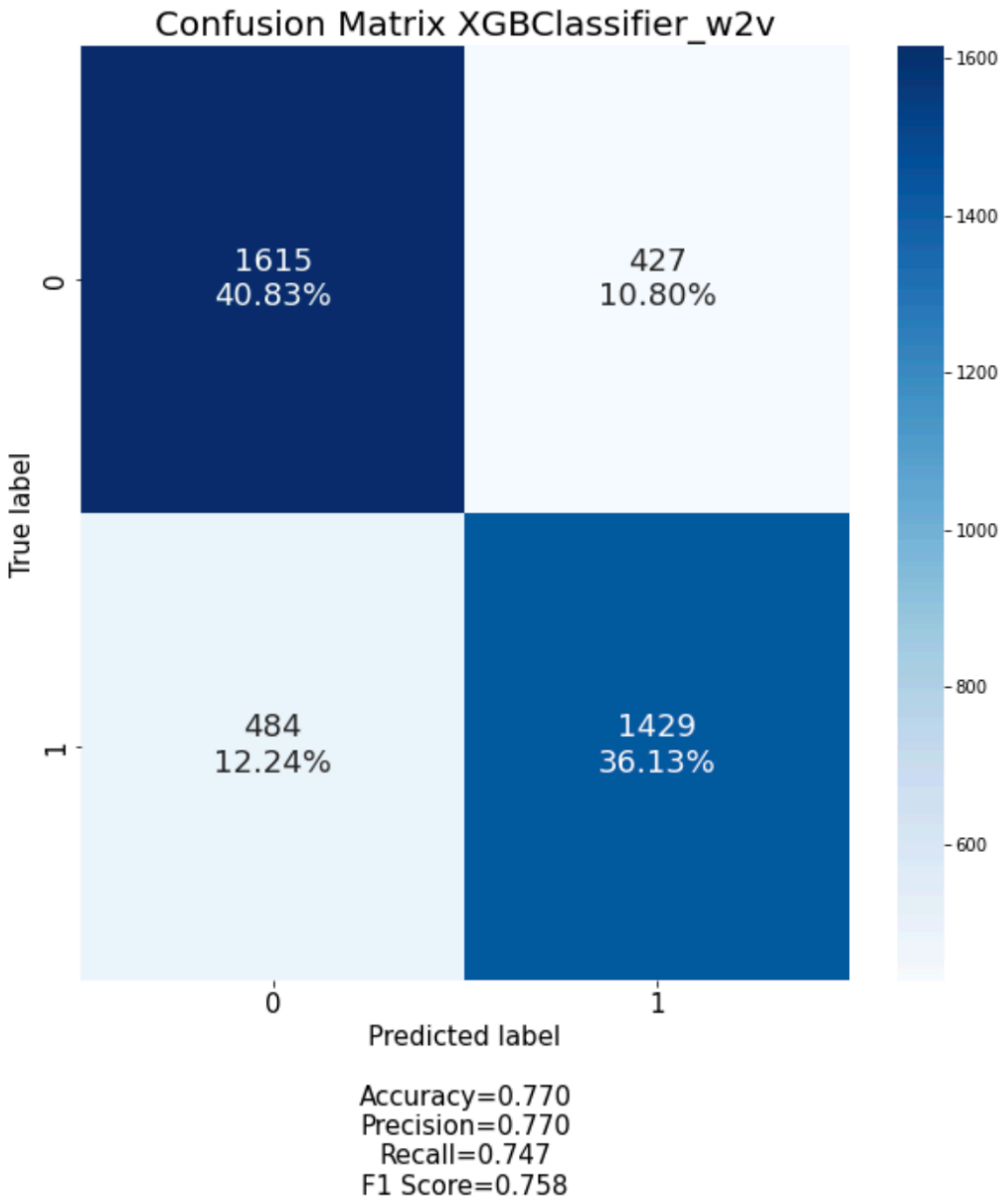
Models

Due to the fact that our data has equal number of Positive and Negative tweets. We're looking for accuracy, also f1score as our evaluation metric. We plot the Confusion Matrix understand if our model is performing on both classification types.

1. Classical approach

For the classical approach we regard the simple machine learning approaches like LogisticRegression, BernoulliNB, RandomForrestClassifier and alos LGBMClassifier and XGBClassifier. The best one was performed here by XGBClassifier trained on word2vec vectorizated data. Let us see the result.

	precision	recall	f1-score	support
0	0.77	0.79	0.78	2042
1	0.77	0.75	0.76	1913
accuracy			0.77	3955
macro avg	0.77	0.77	0.77	3955
weighted avg	0.77	0.77	0.77	3955



Observe a f1 score and the accuracy is equal to 0.77 that is a good result obtained for theese problem.

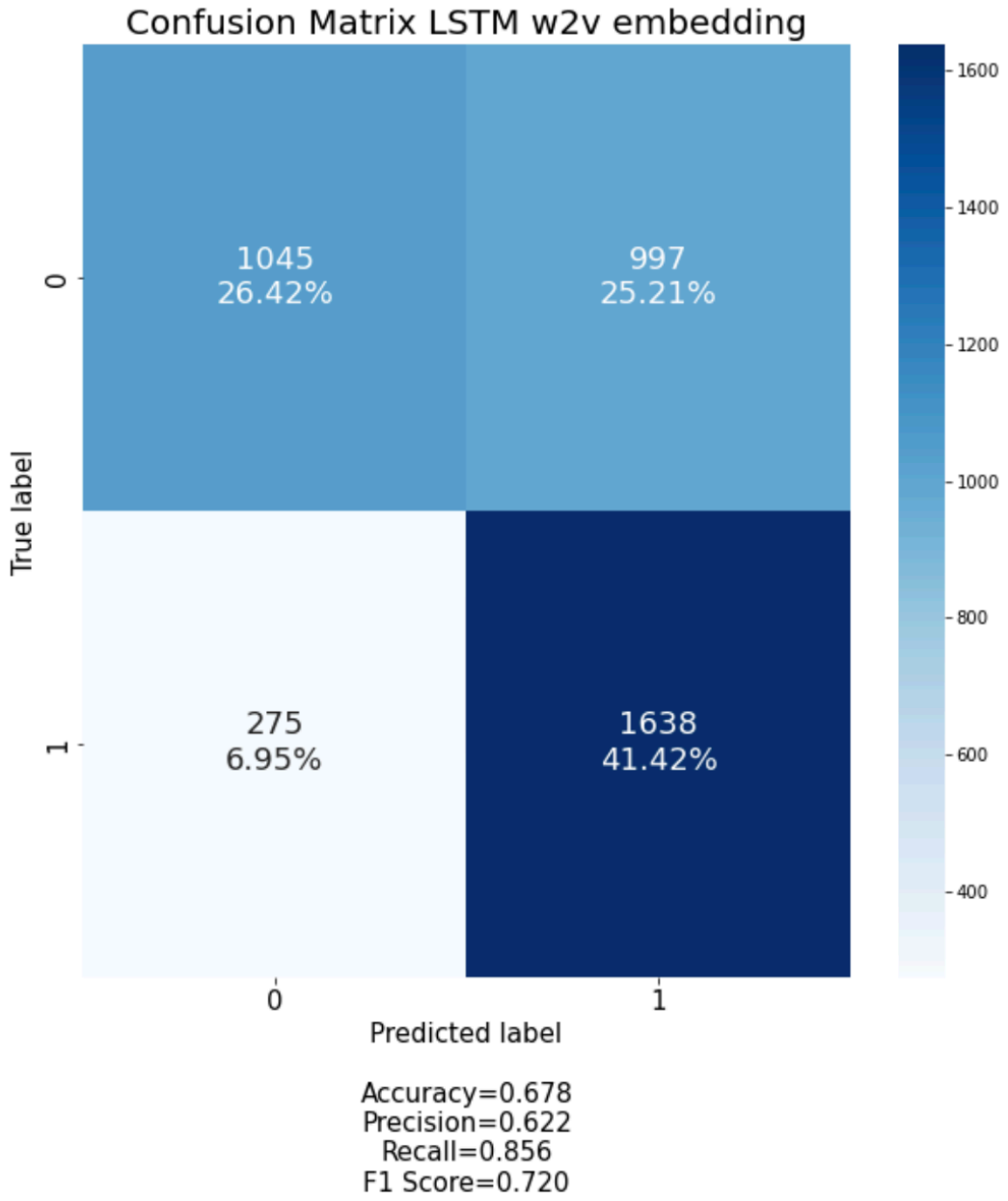
2. Advanced custom models with Neuronal Networks approach using Keras

We are now building our Deep Learning model. While developing a DL model, we should keep in mind of key things like Model Architecture, Hyperparmeter Tuning and Performance of the model. As we saw there can be words predominantly in both positive and negative tweets. This can be a problem by using some simple Machine Learning algorithms like Naive Bayes or SVD, etc. That's why we use Sequence Models. Reccurent Neuronal Network is capable to learn sequence of data and learn a pattern of input sequence to give a sequence or a scalar as output. Our problem is to give a scalar value prediction (either positive or negative tweet)

For this model architecture the following layers. First *Embedding layer* that will generate the embedding layer for each of the input sequence. Next the *Conv1D Layer* that is used to convolve data into smaller feature vectors. Finaly the *LSTM layer* - Long Short Term Memory Layer, that its a variant of RNN which has memory state cell to learn the context of words which are at further along the text to carry contextual meaning rather than just neighbouring words as in case of RNN. We finish the neuronal architecture with a *Dense layer* that is Fully Connected Layers for classification.

Let us see now the results.

	precision	recall	f1-score	support
0	0.79	0.51	0.62	2042
1	0.62	0.86	0.72	1913
accuracy			0.68	3955
macro avg	0.71	0.68	0.67	3955
weighted avg	0.71	0.68	0.67	3955



Unfortunetly, we find a worser results here. This can be due to the fact that a personal computer is used and we made a sample of 10000 of tweets per tweet sentiment. A higher accurecy could be found if learning the model is made on the hole dataset.

Next let's try the BERT models.

3. Advance BERT models

Now we use the Transormer models particularely we will see how BERT (Bidirectional Encoder Representations from Transformers) model works. It’s a bidirectional transformer pre-trained using a combination of masked language modeling objective and next sentence prediction on a large corpus comprising the Toronto Book Corpus and Wikipedia.

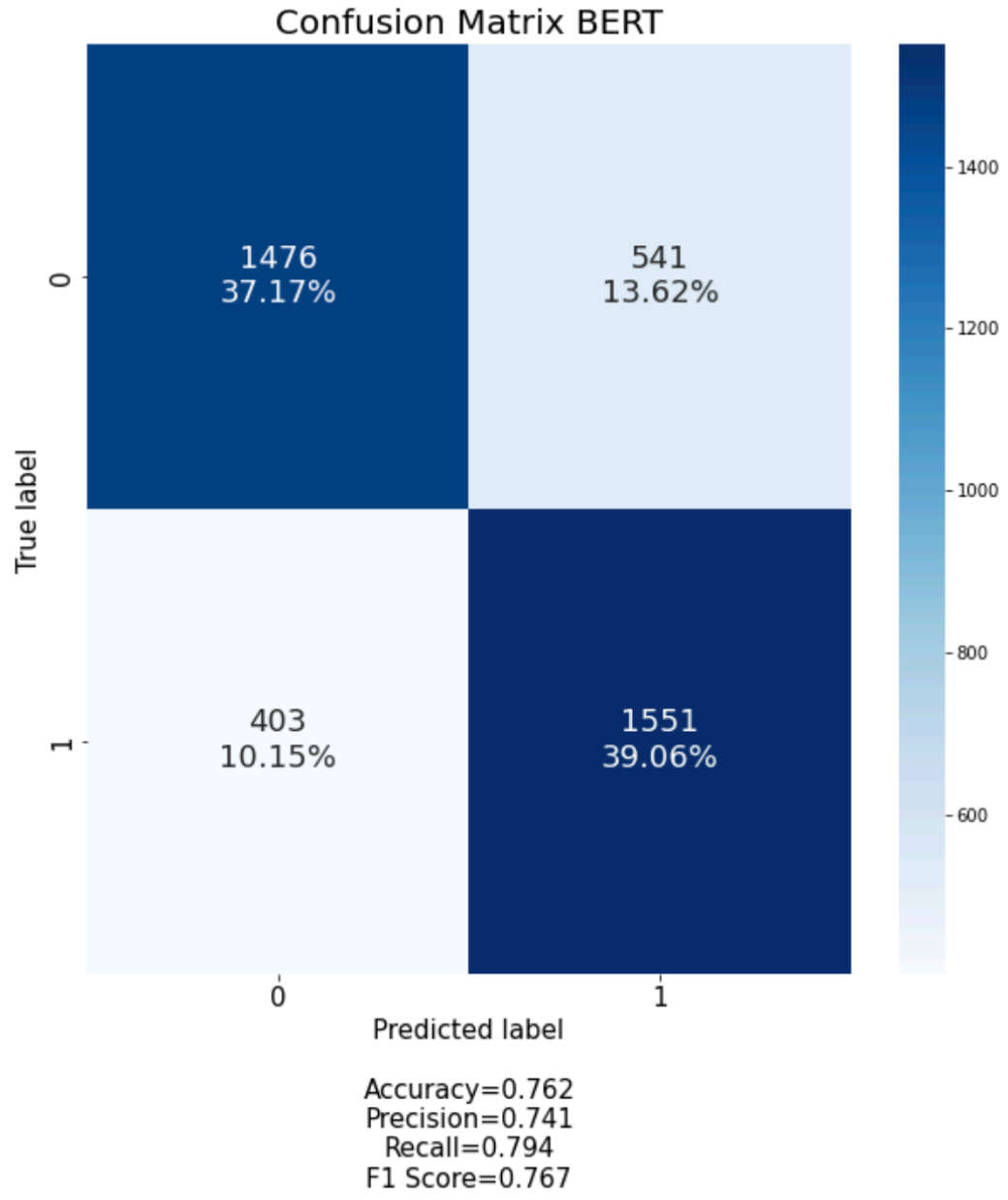
For learning a BERT model first we have to set up a pretrained model. This is loading a pretrained model (we use: *bert-base-uncased*) and setting a BERT tokenizer.

First the Bert Tokenizer, splits the input text into a list of tokens available in the vocabulary. Next we use a pre-trainied model on a bigger dataset and use it as a starting point to learn our smaller dataset. This is called fine-tuning.

Now let us seet the results.

Classification Report

	precision	recall	f1-score	support
0	0.79	0.73	0.76	2017
1	0.74	0.79	0.77	1954
accuracy			0.76	3971
macro avg	0.76	0.76	0.76	3971
weighted avg	0.76	0.76	0.76	3971



Observe that with the BERT model we obtained the best result. The accuracy is 0.762 and the f1_score is 0.767 Also as in the previous section, I suppose that if the model is learned on a hole dataset a better result can be found.

4. Conclusion

In this post we saw three techniques that can be used to predict the sentiment of a tweet. The first one that is the classical approach obtained good results. This is because we took only 20000 tweets as dataset, however in a bigger dataset (as our full dataset containing 1600000 tweets) it can not be the case anymore. Difficulties for simple machine learning techniques can be very fast observed for larger dataset, for exemple when words that are common for the positive and negative tweets. Therefore, Deep Leaning techniques with embedding layer are preferably used for this kind of problem. We called it the "Advanced custom models". Different types of Deep Learning Sequential models are tested, however we saw in the delivered notebook that the one that is the most performant is when using the LSTM model. Next we discover the BERT model that is (according to the most state of the art articles) the most performant. The BERT model has a fundamental advantage, that we do not need to do some text preprocessing. However, the disadvantages of BERT model is that the learning process can take a lot of the ressources (time and storage). The BERT model is preferably learned on a server machine and not on the personal computer.