Text Mining Project – *Emotion Detection and Prediction*

Team: *Sloopy TextMiners*

Diogo Peixoto – 20210993; **Gonçalo Gomes** – 20211007; **João Costa** – 20211005; **Lucas Corrêa** - 20211006

1. Problem Presentation

In this Project, we are dealing with an emotion detection dataset. Each input is a sentence, and the task is to predict the label associated with the emotion expressed in that sentence.

Thus, our goal was to implement a varied set of *natural language processing* (NLP), that could predict, as best as possible, the emotion present in the sentence received as input.

2. Experimental Setup

In this project, there were used several libraries, namely: `nltk` libraries `stopwords`, `WordNetLemmatizer` **;** the `re` library **;** Sklearn' `feature_extraction` libraries `CountVectorizer`, `TfidfVectorizer;` the imblearn's over_sampling packages `SMOTE`, `ADASYN`**;** and finally the sklearn's classifiers GaussianNB, MultinomialNB, BernoulliNB, RandomForestClassifier, SVC, MLPClassifier, KNeighborsClassifier, LogisticRegression, BaggingClassifier.

In order to compare our approaches, the same baseline was used, as suggested in the project specifications: a KNN classifier using the bag of words as text representation and the basic pre-processing of lower case the words with the removal of numbers and special characters. In order to compare, the main classification metrics used were the *accuracy* and the *F1 score*. Although, on some occasions where these two metrics couldn't give provide the distinction needed among models, *recall* and *precision* of each were used as a second tier of comparison metrics.

3. Best Approach

For the sake of a better description, the best approach taken is described in three parts: preprocessing, feature extraction and model implementation.

Regarding the **preprocessing**, the steps adopted on our best approach were the result of discussion and testing, until a final function was created called "*preprocessing_tok1*". As an example, it receives a text ("*I'm too old to be traded in*") and outputs the following tokens list ("['old', 'traded']").

With regards to the function itself, the first step was to remove all English stop words, except for "not", which appear in the corpus 584 times. Since the removal of "*not"* in a sentence can change the meaning of a sentence or of an expression, we decided to maintain it [1]. Then we proceed to lowercasing and removal of punctuation, as well as the removal of the following tokens ['LOCATION'] and ['PERSON'], which appear 306 and 1485 times, respectively. The decision to remove these tokens were made after noticing that the removal would lead to a marginal, but yet positive improvement on the overall accuracy of the final model. As the fourth and fifth step, the strings were split into a list, which were posteriorly lemmatized.

In the feature extraction, our group decided to perform *Term Frequency – Inverse Document Frequency* (TF-IDF). We applied sklearn's `feature.extraction.text.TfidfVectorizer` to perform this step. In order to optimize it, several parameters were applied to this function:

- *`max_features`* = 10000 and `max_df` = 0.9. With *`max_features`* we limited the analysis to the 10.000 more frequent words, and in the combination with `max_df`, we ignored terms

that had a document frequency higher than 90%. The intuition is to limit the feature extraction to frequent, but still distinctive-worth words.

- `ngram_range` = (1,3). By extending the tuple 's upper limit to 3, we allowed the vectorized to consider uni, bi and trigrams during the feature extraction. The intuition is that some combinations of words mean different things, than the words separately
- `tokenizer = preprocessing_tok1`. Lastly, the previous design preprocessing function was parsed as into the tokenizer argument, allowing the execution of the TF-IDF while presenting the pre-processed text.

In the model implementation step, the chosen algorithm was a Support Vector Classifier (SVC) algorithm using a linear kernel. This was the result of extensive parallel testing with different algorithms, in which it outperformed the other models in accuracy and recall.

Small variations of this model were tested: using two different oversampling techniques to tackle the imbalance label problem, namely SMOTE and ADASYN; introducing the Part of Speech Tagging (POS) in the preprocessing function; increasing the `max_feature` of the TF-IDF vectorizer to 20000; or even changing the C parameter of the SVM. Despite all these variations, none had proved to increase the performance of our best approach model described above.

4. Evaluation of the Best Approach

The results on the validation set are summarized below on table 1. Only the accuracy metric is presented to ease the interpretation.

MultinomialNB	**BernoulliNB**	**Random Forest** (max_depth=5)	**SVC** (kernel=linear)	**MLP** (5,5,3)	**Logistic Regression**
0.35	0.29	0.27	0.36	0.26	0.35

Table 1: Accuracy metric results for different algorithms applied to the validation set

The best result was achieved with SVC with linear kernel. Firstly, looking into its classification report on figure 2, it is noticeable the worst f1-score results are for the classes 3,4,6 and 7.

	precision	recall	f1-score	support
Fear	0.57	0.31	0.40	388
Trust	0.31	0.26	0.28	203
Surprise	0.05	0.16	0.08	25
Anticipation	0.15	0.25	0.19	65
Anger	0.27	0.34	0.30	77
Disgust	0.03	0.09	0.05	33
Sadness	0.21	0.19	0.20	108
Joy	0.16	0.25	0.19	101
accuracy			0.27	1000
macro avg	0.22	0.23	0.21	1000
weighted avg	0.36	0.27	0.29	1000

Figure 1 – Baseline classification report on the validation set

	precision	recall	f1-score	support
1	0.33	0.65	0.44	211
2	0.42	0.42	0.42	170
3	0.20	0.10	0.14	77
4	0.30	0.23	0.26	104
5	0.43	0.38	0.40	97
6	0.37	0.20	0.26	87
7	0.37	0.16	0.22	96
8	0.40	0.31	0.35	158
accuracy			0.36	1000
macro avg	0.35	0.31	0.31	1000
weighted avg	0.36	0.36	0.34	1000

Figure 2 – SVC Classification report on the validation set

Looking at the training label counting from the dataset in figure 3, we do understand the worst results for the classes mentioned above are related with the least training data available. With the presence of an unbalanced dataset, we have decided to try two oversampling techniques, the SMOTE and the ADASYN. ADASYN has provided the best results, which are presented in figure 4. It is clear that the f1-score for the classes referred have improved, but overall, the accuracy decreased to 0.33. Therefore, ADASYN was not considered in the final approach.

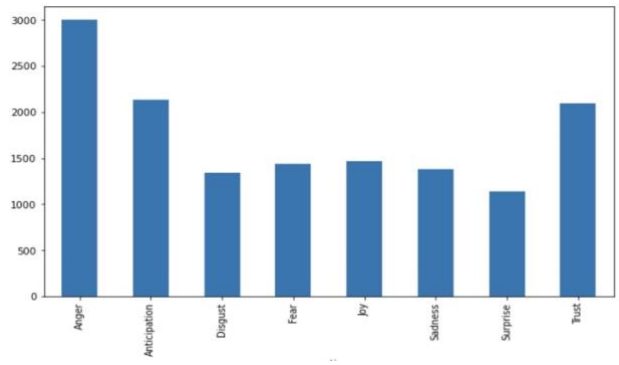

Figure 3 – Emotion label counting

	precision	recall	f1-score	support
1	0.41	0.37	0.39	211
2	0.41	0.34	0.37	170
3	0.23	0.22	0.23	77
4	0.28	0.31	0.29	104
5	0.40	0.46	0.43	97
6	0.28	0.28	0.28	87
7	0.21	0.38	0.27	96
8	0.40	0.27	0.32	158
accuracy			0.33	1000
macro avg	0.33	0.33	0.32	1000
weighted avg	0.35	0.33	0.34	1000

Figure 4 – SVC with ADASYN Classification report on the validation set

It is important to note that both the oversampling techniques and the POS have been applied to all algorithms presented in table 1, but in none of them the accuracy got beyond the 0.35. Based on that, we got the certainty our best algorithm, out of the ones tried, was the SVC one.

With that in mind, we have tried to further fine-tune the results around this single algorithm. We have experimented with different regularization parameters, within the SVC algorithm itself, and used a different max number of features within the TF-IDF vectorizer. Anyway, the results did not improve compared to when using the default regularization parameter defined by the SVC scikit learn library or with the max-features value equal to 10000 on the TDF-IDF technique, which was our first iteration, based on the value used in the practical text mining classes.

We have still experimented using the SVC with a non-linear kernel, but not only the computational time increased drastically, but also the chances of overfitting and not being able to generalize to the test set, led us to not further investigate this option.

Therefore, the best approach as presented before has outputted the results in figure 2. The main concern is related to the poor results for the minor classes. They state the importance of having enough training data for the machine learning models being able to learn sufficiently. On the positive side, we highlight the results improvement achieved starting from the baseline.

5. Future work

After this work development, we believe that it would be interesting, in the future, to apply some other methodologies, in order to try enhancing the results obtained.

One of the procedures that we could adjust is with regards the stop words removal process. In this project, we have decided to only remove the word “not” as per the justification presented above. During the project development, we have argued about removing the most frequent stop words present in our corpus and keeping the ones least frequent, limited by a frequency threshold of 100 occurrences, for instance. The justification lies on the higher classification discriminatory power of the least frequent words. However, we were afraid these least frequent stop words could be much more frequent on the test set, by the nature of the stop words itself. If it would happen, it would introduce bias to our model. Therefore, we have not moved ahead with this option, but we find interesting to try it in the future.

Another possible future approach would be, for example, to create a function that would allow us to eliminate non-English words, since they do not contribute positively to a better performance of the model.

Finally, another possibility would be to apply Bert, which is a pre-trained NLP model that managed to achieve the best accuracies for some of the NLP tasks in 2018 [2]. However, this approach would be outside of the scope of this project and, therefore, we have decided not to test it.

References

[1] – Khanna, Chetna – "*Text pre-processing: Stop words removal using different libraries*" – Towards Data Science – Dec 6, 2019 – Available at: https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a

[2] –Yalçin, Orhan G. – "*Sentiment Analysis in 10 Minutes with Bert and TensorFlow*" – Nov 28, 2020 – Available at: https://towardsdatascience.com/sentiment-analysis-in-10-minutes-with-bert-and-hugging-face-294e8a04b671