EMOTION RECOGNITION USING TEXT ANALYSIS

Group ACFS

Ana Luís, m20210671 | Carolina Machado, m20210676

Francisco Calha, m20210673 | Sara Arana, m20210672

1. Description of the best approach:

- **The preprocessing applied**

On a first instance, we replaced all the « ' » for a blank space, i.e., 'He'll be safe with me' transformed to 'He ll be safe with me'. This step was done because when removing the stop words, these cases were not taken into consideration. Afterwards, all punctuation was removed, excepting for the « ! » and « ? » signs, since, sometimes, these express a sentiment. Sometimes, when words and numbers are combined in sentences, it can create a problem for machines to learn patterns, so, our next step was to remove all the numbers in the sentences. However, we also tried an approach by which we replaced them by the token [DIGIT]. Then, we transformed all words to lowercase because it is easier for the machine to learn if the words are all in the same format (because lower and upper case words are treated differently). This was followed by the removal of all the stop words – which are commonly occurring words that do not provide any valuable information. Then, we applied Lemmatization. Stemming was also an experimented approach, but it did not improve our results. Afterwards, we checked for empty sentences which we decided to remove only for the train set to better train our model. This step was done because the empty sentences can express different emotions and the model could be confused and not learn properly the patterns. In addition, we notice the existence of 116 sentences with only one character, which are, on most of them, a « ! » or « ? » and only one sentence with « u ». We figured it would be helpful to remove them, but by doing so our model performance decreased, both in the learning process and in terms of the accuracy of the test set. So, we ended up maintaining the sentences containing « ! » and « ? », since they proved to be meaningful to our emotion recognition model. We noticed that both characters are related to all emotions, but « ? » is most related to emotions 2 and 7 and « ! » is more related to emotion 1. The sentence with « u » was removed because it was neither meaningful nor relevant in our dataset.

- **The features extracted from the sentences**

To vectorize the sentences in our dataset, several approaches were explored, namely: Bag of Words (CountVectorizer, Bigrams, Trigrams), TF-IDF, and Word Embedding (Word2Vec).

TF-IDF was the feature extraction method used in our best approach. In fact, we had already expected it could be one of the best approaches, particularly, when comparing to Bag of Words, because, unlike those methods, it attributes a certain importance to the word when vectorizing it. That is, it gives a higher weight to words that are unique to a particular document compared to words that are used commonly across documents. In addition, we were also curious to see how Embedding Methods perform in our data. For that, we choose to also explore Word2Vec, one of the most known methods in this field. This method is quite interesting since each word is encoded, while considering its context on the document, semantic and syntactic similarity, and by finding relations with the other words.

- **The model/approach implemented**

The classifier from which we obtained a better score for our problem was the Multinomial Naïve Bayes. This classifier was initially chosen for being one of the most used for multi-label text classification and for the fact that it is relatively fast. This model uses the Bayes theorem and has two assumptions, namely, each feature is independent from each other and each one has the same importance for the outcome. In fact, one advantage of Multinomial Naïve Bayes is its capacity to classify records using a relatively small corpus, which is our case.

This model proves to be useful in classification problems using discrete features, for example, for word counts in text classification, so we expected good results when combining IT with Count Vectorizer. Theoretically, this multinomial distribution requires, usually, feature counts of integer values. However, in practical terms, it can also perform well with fractional counts such as TD-IDF, which is what happened in our case.

For the development of our model, several other classifiers were applied to our data – KNN with cosine similarity, multinomial Logistic Regression, linear SVM, some ensemble classifiers (Gradient Boosting, Random Forest, and Stacking) and Neural Networks (CNN). These classifiers were chosen for their known good performance in text analysis classification problems and for our curiosity on how they would behave in the problem we were dealing with. In addition, and to achieve better results, a Grid Search, for hyper-parameter tunning, was performed.

2.Experimental setup:

The libraries used, in the development of this project, were the following: pandas, numpy, matplotlib, seaborn, warnings, nltk, string, sklearn, keras and tensorflow.

- **How the approaches were evaluated:**

To evaluate the different classifiers tested, the metric we used was the F1 score (micro). This metric is commonly used to determine the quality of multi-classification problems, and *«it measures the aggregated contributions of all classes»*. For this type of problems, it is useful to use micro-averaging because it gives more importance to common labels since it provides each sample the same importance. In our case, this metric is the same as the Accuracy score, because each record is only related to one class. For our problem, a weak baseline could be a KNN with Count Vectorizer and the pre-processing done above, and a strong baseline could be a linear SVM with Count Vectorizer and the same pre-processing, once again. Our best approach was able to achieve better results than both baselines, as we will see in the following section.

3. An evaluation of the best approach:

Now, we will present the results of our best approach, by providing a comparison to both baselines mentioned previously.

Classification Report – Strong baseline (svm.SVC(C=0.7, gamma='auto', kernel='linear') + CountVectorizer)

	precision	recall	f1-score	support
1	0.31	0.59	0.41	211
2	0.36	0.34	0.35	170
3	0.22	0.14	0.17	77
4	0.38	0.26	0.31	104
5	0.41	0.37	0.39	97
6	0.37	0.23	0.28	87
7	0.28	0.18	0.22	96
8	0.39	0.30	0.34	158
accuracy			0.34	1000
macro avg	0.34	0.30	0.31	1000
weighted avg	0.35	0.34	0.33	1000

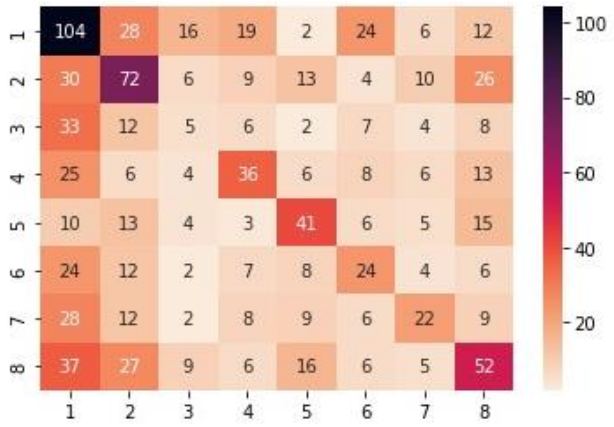
Classification Report – Weak baseline (KNeighborsClassifier(metric='cosine', n_jobs=-1, n_neighbors=22,weights='distance') + CountVectorizer)

	precision	recall	f1-score	support
1	0.31	0.55	0.39	211
2	0.31	0.31	0.31	170
3	0.19	0.05	0.08	77
4	0.30	0.24	0.27	104
5	0.38	0.36	0.37	97
6	0.32	0.14	0.19	87
7	0.23	0.20	0.21	96
8	0.31	0.25	0.28	158
accuracy			0.30	1000
macro avg	0.29	0.26	0.26	1000
weighted avg	0.30	0.30	0.29	1000

Classification Report – Best approach
(MultinomialNB(fit_prior=False) + TF-IDF)

	precision	recall	f1-score	support
1	0.36	0.49	0.41	211
2	0.40	0.42	0.41	170
3	0.10	0.06	0.08	77
4	0.38	0.35	0.36	104
5	0.42	0.42	0.42	97
6	0.28	0.28	0.28	87
7	0.35	0.23	0.28	96
8	0.37	0.33	0.35	158
accuracy			0.36	1000
macro avg	0.33	0.32	0.32	1000
weighted avg	0.35	0.36	0.35	1000

Confusion Matrix – Best model (MultinomialNB(fit_prior=False)



From the justifications presented previously, our best model at classifying emotions was the **Multinomial Naïve Bayes**. This model was not the only one that surpassed the value from the strong baseline presented but it was the one that obtained a higher accuracy score, of 0.36, which means it correctly predicted 36 percent of our test set. In addition, the macro average Precision is higher in the strong baseline than in our best approach. In the strong baseline, according to the Precision per label, we can also see that it has a higher value in the minority classes. This means that out of all the observations classified as belonging to these classes, the actual number of observations being of this class is higher in the strong baseline. Regarding the recall, according to the macro average, our best model obtained a better score than the strong baseline. This means that, overall, the best model is learning better how to identify the classes. Nevertheless, the strong baseline is better at identifying records from classes 1 and 3. Regarding the F1 score, both macro and per label, the best model has a higher score than the strong baseline, except in the class 3.

From the Confusion Matrix, we can verify that almost all observations from class 3 are being misclassified. This might be due to the fact of this class is one of our smallest classes, in terms of observations, or this emotion is really hard to classify with the records provided. However, this pattern was verified in all other presented models. It is also visible that class 1 is the one with the most correctly classified observations. This might be happening since it is the class with the highest number of observations. Another interesting result is the fact that the number of correctly classified observations from class 6 is the same as the number of misclassified observations labeled as class 1. This might mean that there is a certain similarity between these two emotions, in terms of their recognition through text analysis.

4.Future work

Overall, our model obtained good results, considering the data we had access to. Nevertheless, there are some steps we could have done to improve the performance of our project, namely:

- Further explore the ensemble classifiers (Gradient Boosting, Random Forest, and Stacking), by applying a more complete grid search to find the best parameters. We have good expectations mainly for Stacking, since it combines several classifiers, namely the ones mentioned above. The results obtained with it were already good, so further exploring this approach could lead to a better final score.
- We considered at the beginning that our dataset was quite balanced and for that reason resampling techniques were not tested. However, with the confusion matrix of our best models and with the Recall values presented above, to correctly classify an observation from class 3 was a challenge for the models trained and one possible reason might be related to the fact that it is one of our smallest classes, in terms of size.
- Finally, although CNNs usually perform better in Speech Emotion Recognition tasks than RNNs, we could also try to explore it. Or even further explore and extend the CNN experimented.

5.Bibliography

Naïve Bayes: https://towardsdatascience.com/multinomial-na%C3%AFve-bayes-for-documents-classification-and-natural-language-processing-nlp-e08cc848ce6

F1 Score (micro): https://peltarion.com/knowledge-center/documentation/evaluation-view/classification-loss-metrics/micro-f1-score

N-Grams: https://www.analyticsvidhya.com/blog/2021/09/what-are-n-grams-and-how-to-implement-them-in-python/

Multinomial Logistic Regression: https://machinelearningmastery.com/multinomial-logistic-regression-with-python/

Emotion Recognition Classification: https://medium.com/geekculture/simple-emotion-classification-in-python-40fb24692541, https://betterprogramming.pub/unlocking-emotions-in-text-using-python-6d062b48d71f

https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf

https://web.stanford.edu/~jurafsky/slp3/5.pdf