Text Mining Project

Inês Ribeiro m20210595 **José Dias, m20211009** **Matias Neves, m20211000**

Group name: IJM

1. Introduction

This project is for the Text Mining course and the goal is to predict the emotion displayed in a given sentence (8 different labels/emotions).

There are 14.000 sentences in the training set and 1.000 in the validation set. Through pre-processing, the words used as features in this project were obtained from these sentences. During the project, multiple approaches were used in order to get the best score possible.

Google Colab was used to do this project, alongside the following python libraries: pandas, numpy, nltk, keras and sklearn.

2. Pre-processing

For the pre-processing, multiple functions were used to work on the sentences, and present them in a better way to the final model.

The pre-processing tasks that were performed were: remove punctuation, tokenize, remove larger words (models were compared with and without removing words with 3 characters), remove stop words, stemming and lemmatization.

3. Best Model

The best model was a deep learning model and it was a Bidirectional LSTM model.

To train this data in a deep learning model, it was required to tokenize it and also to use the pad_sequences method (transforms a list of sequences into a 2D Numpy array). It was also necessary to perform encoding of the labels.

The pre-processing of the best model did not include removing words with 3 characters, because it achieved a better score without removing them.

For this model, word embeddings were used, more specifically GloVe embeddings (algorithm that obtains vector representations for words. Training is performed on aggregated global word to word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space).

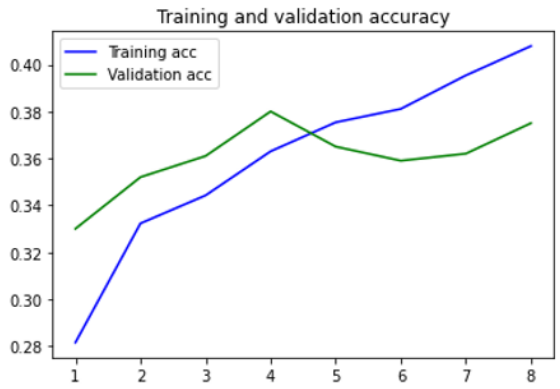
To be considered a good model, it is necessary to look at the overfitting, and at the validation accuracy.

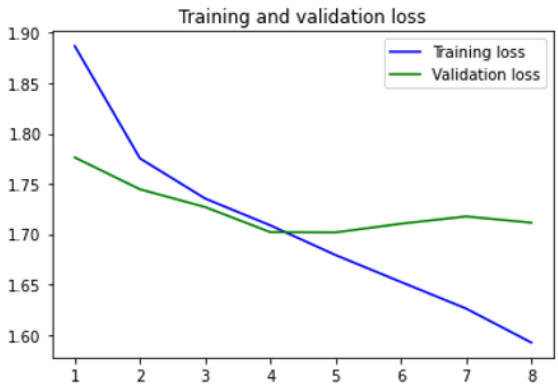
This model was the one that presented the highest score with the least overfitting to the training data. The final score of the model was a 0.4079 accuracy on the training data and a 0.3750 accuracy on the validation data. The recall of the best model was 0.12, the precision 0.58 and the F1 score was 0.20 (0.44 for Anger, 0.43 for Anticipation, 0.24 for Disgust, 0.30 for Fear, 0.44 for Joy, 0.29 for Sadness, 0.35 for Surprise and 0.32 for Trust).

To train the model, earlystopping was used, what a patience of 3, to avoid continuing the training if the scores are no longer improving.

A grid search was performed to find the best hyperparameters. However, the final model was not the model chosen by the grid search, because the model with index 3 had a smaller overfitting with the same validation accuracy than model with index 2 (the one automatically chosen by the grid search).

In the grid search, not all parameters were used because of GPU limitations in Google Colab. Only 3 parameters were used in the grid search, all dropouts, and if there were no GPU limitations, a bigger grid search could be performed, and as a consequence, a better score could probably be achieved.





4. Performance Results

Besides being the model with the best performance on validation data, this model didn't show big signs of overfitting, with train and validation scores being very similar in all epochs, and loss being very similar between training and validation data.

The accuracy of this model is 0.38, or 38%.

From the F1-scores of each class, it's seen that the model was slightly better at detecting "anger", "anticipation", and "joy" (the latter of which being a surprise since, unlike the other 2 classes, it didn´t have a relatively big support) and was particularly poor at detecting "disgust".

The confusion matrix showed some interesting data. Such as the amount of words correctly associated with "disgust" being smaller than both the amount of words incorrectly associated with "disgust", and the amount of words that belong to the "disgust" class but were incorrectly associated to the "anger class". There was also a relatively big amount of words incorrectly associated to "anger", "anticipation", or "joy", that belong to the "trust" class.

Classification Report

	precision	recall	f1-score	support
Anger	0.41	0.49	0.44	211
Anticipation	0.43	0.44	0.43	170
Disgust	0.25	0.22	0.24	77
Fear	0.37	0.25	0.30	104
Joy	0.37	0.55	0.44	97
Sadness	0.33	0.25	0.29	87
Surprise	0.31	0.40	0.35	96
Trust	0.39	0.27	0.32	158
accuracy			0.38	1000
macro avg	0.36	0.36	0.35	1000
weighted avg	0.37	0.38	0.37	1000

Confusion matrix

```
[[103  25  19  11   7  11  20  15]
 [ 29  74   8   6  20   2  18  13]
 [ 18   8  17  10   6   6   3   9]
 [ 25  12   5  26   5   9  11  11]
 [  8   7   3   2  53   5  11   8]
 [ 18   9   5   5  10  22  14   4]
 [ 17  10   4   5  12   5  38   5]
 [ 34  29   6   5  29   6   7  42]]
```

5. Baselines

The weaker baseline is KNN with Count Vectorizer and achieved an accuracy score of 0.26. The stronger baseline is a Connect model with an accuracy score of 0.3250.

6. Other approaches

Other approaches were used during the development of this project.

6.1. Pipelines

In total, 91 models/pipelines were created to check if any yielded good results. Multiple combinations were used namely, undersampling (Tomek Links), oversampling (SMOTE), Count Vectorizer, TfidfVectorizer and TfidfTransformer.

Tfidftransformer and Tfidfvectorizer aim to do the same thing, which is to convert a collection of raw documents to a matrix of TF-IDF features. The only difference is that with Tfidftransformer, it will systematically compute the word counts, generate idf values and then compute a tfidf score or set of scores.

The models used in those pipelines were: MultinomialNB, SGDClassifier, LogisticRegression, MLPClassifier, GradientBoostingClassifier, RandomForestClassifier, DecisionTreeClassifier, ExtraTreesClassifier, XGBClassifier, AdaBoostClassifier, SVC, KNeighborsClassifier and PassiveAggressiveClassifier. The scores of the pipelines were not that great and had a lot of overfitting.

The highest score achieved with the pipelines was a 0.361 validation accuracy (with 0.5095 training accuracy), and it was in model 16 – Count Vectorizer and Logistic Regression (without removing small words in the pre-processing).

6.2. Deep Learning

Other deep learning models were used during the elaboration of the project, but all achieved worse results.The models were: RNN with LSTM, ConvNet, CNN + Bidirectional GRU, LSTM with Attention. LSTM stands for long short-term memory, and they are capable of learning long-term dependencies, especially in sequence prediction problems. It has feedback connections, which means it can process the entire sequence of data.

7. Limitations and future work

As previously mentioned, one of the limitations was the GPU of Google Colab. It was limited in terms of the time the GPU could be used, and to run grid search the GPU was required, otherwise it would take days. Therefore, the approach that was taken was to reduce the number of parameters to only 3, and the grid search took around 4 hours with the usage of GPU.

For future work, there are some things that could be done. Other methods of under sampling and over sampling could be used, and maybe achieve better results. Some of those methods also take a lot of time to train. Other models could also be explored, especially pre trained models with different word embeddings (for example the Bert model). The last thing to mention for future word is to try a different pre-processing. Different dataframes could be used for different types of pre-processing and those could be added to the pipelines (with different punctuation removal and stop word removal).

8. Bibliography:

https://keras.io/api/layers/recurrent_layers/lstm/

https://intellipaat.com/blog/what-is-lstm/

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html

https://github.com/kavgan/nlp-in-practice/blob/master/tfidftransformer/TFIDFTransformer%20vs.%20TFIDFVectorizer.ipynb

https://github.com/Vish4github/Transfer-Learning-in-Text-Classification/blob/master/News%20Category%20Classification%20Using%20Deep%20Learning.ipynb