TEXT MINING

OPINION MINING PROJECT, USING XED DATASET FOR MULTICLASS EMOTION DETECTION

PUZZLES Team: António Cymbron (r20181059), **Duarte Redinha** (r20181066), **Maria João Marques** (r20181119)
MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS, WITH A SPECIALIZATION IN DATA SCIENCE
NOVA INFORMATION MANAGEMENT SCHOOL

I. CONTEXT & INTRODUCTION:

According to Oxford Languages, Opinion Mining consists of identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral.

For this purpose, a subset of the XED dataset for multiclass emotion detection, which consists of English and Finnish movie subtitles from OPUS corpus, annotated with emotional polarity and discrete emotions, will be used.

II. EXPERIMENTAL SETUP:

Our setup initiates with the usage of Python 3, Machine Learning models and multiple available libraries, namely *pandas*, *numpy*, *matplotlib*, *google*, *tqdm*, *copy*, *nltk*, *bs4*, *re*, *sklearn*, *imblearn*, *tensorflow*, *warnings* and *transformers*. Also, to improve our productivity as a group, we resorted to tools such as *Google Colab*, and imported the available data accordingly to this tool.

Before starting to develop candidate models for the weaker and stronger baselines, we decided to firstly understand better our data. For that matter, we defined several functions to use that would provide us useful information – *label_counter*, that counts the number of sentences labelled with each emotion; *word_counter*, that provides the frequency of each word; and *avg_unique_words_per_emotion*, that provides the average of unique words per sentence for each of the emotions. These functions were used before and after the pre-processing phase, which gave us valuable insights of how our data was evolving. After this, we resorted once more to functions to develop a pipeline for pre-processing, *text_preprocessing*, that contains actions to lowercase, remove tags, punctuation, stop words and multiple spaces, replace numbers and the characters '?', '!' by tokens. It is worth mentioning that we decided to replace each of the characters with a different token – *[NUMBER]*, *[INTERROGATION]*, and *[EXCLAMATION]*, respectively. This decision was based on the belief that, handling a Sentiment Analysis project, that aims to distinguish several emotions, replacing both of the last characters with the same token would prejudice the performance of the model.

After this, we developed another pipeline, *pipeline_data*, this time for performing feature extraction. Inside it, we used the pre-processed corpus that allows to apply the Bag-Of-Words model, a representation of text that describes the occurrence of words within a document (Brownlee, J., 2017), TF-IDF, a technique to quantify words in a set of documents, where a score is computed for each word to signify its importance in the document and corpus (Scott, W., 2019), Lemmatization, that aims to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the *lemma* (Stanford NLP Group, 2009), or Stemming, a crude heuristic process that chops off the ends of words, and often includes the removal of derivational affixes (Stanford NLP Group, 2009), according to the user needs. This pipeline is also prepared to be applied to all three datasets provided. Before any model construction, our group also decided to analyse the top *N-grams* in the *corpus* – unigrams, bigrams and trigrams -, by plotting their frequencies.

All these actions related to getting to know our *corpus* better and doing pre-processing were crucial to the elaboration of our models. The metrics evaluated throughout the models' work were Precision, Recall, F1-Score (per label and macro-average), and Accuracy for each. However, for the models that are neither our baseline, nor our best approach, only the development Accuracies will be presented in this final report. Nevertheless, for further information, the classification reports (arranged using *sklearn*) for each intermediate approach are available in the provided code.

Table 1 *- Baseline considered for the Opinion Mining Project (Bag-Of-Words model + K-Nearest Neighbours (n=7))*

Baseline BOW + KNN(7)			
	Precision	**Recall**	**F1-Score**
1: Anger	0,51	0,33	0,4
2: Anticipation	0,31	0,29	0,3
3: Disgust	0,09	0,16	0,11
4: Fear	0,21	0,25	0,23
5: Joy	0,28	0,3	0,29
6: Sadness	0,17	0,22	0,19
7: Surprise	0,23	0,27	0,25
8: Trust	0,26	0,33	0,29
Accuracy	-	-	**0,29**
Macro Avg	0,26	0,27	0,26

Table 2 *– Intermediate Approaches for the Opinion Mining Project (Model Informations and Accuracy, for each)*

Model Informations					Accuracy
Model	**Vectorizer**	**SMOTE**	**Lemmatization**	**Stemming**	
KNN	BOW	No	No	No	**0,29**
KNN	TF-IDF	No	No	No	**0,32**
NB	BOW	No	No	No	**0,36**
NB	TF-IDF	No	No	No	**0,33**
SVC	BOW	No	No	No	**0,36**
SVC	TF-IDF	No	No	No	**0,36**
SVC	TF-IDF	Yes	No	No	**0,35**
SVC	TF-IDF	Yes	Yes	No	**0,35**
SVC	TF-IDF	Yes	No	Yes	**0,34**
BERT	-	-	-	-	**0,51**
BERT (with class_weights)	-	-	-	-	**0,48**

Regarding the model considered as our baseline, we designed a simplistic approach consisting of applying the Bag-Of-Words model, followed by a Machine Learning model, namely the K-Nearest Neighbours, employed with 7 as the number of neighbours considered. This approach, whose metrics can be consulted in Table 1, reached an Accuracy of 0,29 on the development set. The label that was best predicted by the model was the Anger one, and the worst predicted one was the 3rd one, referring to the emotion of Disgust. These results are probably due to the fact that we are in the presence of an unbalanced dataset, being the 1st emotion the one most represented on the train dataset – with 2999 sentences - and being the 3rd emotion the second least represented one on the same dataset – with only 1343 sentences, according to the *label_counter* function defined in the beginning. This information made us evolve our models, namely by applying SMOTE due to the last conclusion exposed. However, priorly to implementing this oversampling approach, we applied the same Machine Learning model with Term Frequency-Inverse Document Frequency (TF-IDF), a more powerful model for feature extraction than Bag-Of-Words, and the results gotten were better (0,32). The same reasoning was used for the several other Machine Learning models tried for this task, namely Naïve-Bayes, and Support Vector Machines (with a Radial Basis Function (RBF) kernel, which showed to be better than using a Linear kernel, thanks to a *GridSearch,* provided by the *sklearn* package). Being so, for the three presented algorithms, we applied both Bag-Of-Words and TF-IDF. These formed our intermediate approaches, whose metrics can be consulted in Table 2.

Finally, to the best of our intermediate approaches - Support Vector Classification with TF-IDF -, we applied the Synthetic Minority Over-sampling Technique, or SMOTE, as initially planned. By implementing this technique, we aimed to perform over-sampling in an artificial way, to balance the uneven number of sentences by emotion. It is worth mentioning that the selected approach for trying SMOTE was with TF-IDF and not BOW, since the first model presented more positive values for the remaining metrics being considered, namely the macro average for Precision (0,31 > 0,30), Recall (0,37 > 0,36), and F1-Score (0,32 > 0,30) metrics. Once SMOTE was applied, we noticed a lower Accuracy score, that the SVC+TF-IDF without SMOTE model. However, we immediately noticed that the previously worst predicted emotions, were now better predicted, which we had into account, since this option would clearly train better our model. Afterwards, we also tried to apply Lemmatization and Stemming to the pre-processing of this resulting model. However, none of them produced the wanted results, leading us to choose, at this stage, a model without any of these two pre-processing steps.

III. OUR BEST APPROACH:

After achieving what we believe were the best possible results with the previous models and tools, we aimed for higher approaches and used a new language representation model called BERT, or Bidirectional Encoder Representations from Transformers. According to Horev, R. (2018), this model is a Neural Network that arose from a recent paper published by researchers at Google AI Language, and its key technical innovation is applying the bidirectional training of *Transformer*, a popular attention model, to language modelling. The paper's results show that a language model which is bi-directionally trained can have a deeper sense of language context and flow than single-direction language models. Knowing this, the pre-processing

and feature extraction of the model was done by lowercasing and tokenizing the sentences using *WordPiece* and a vocabulary size of 30,000.

To implement this model we started by transforming all labels with the value 8 into 0, because had in mind that since we were working with a Neural Network, the output layer would need to have eight neurons, one for each layer. Regarding the chosen tokenizer, we used the *bert-base-uncased*, since this pre-trained model is suitable for our *corpus* due to the fact that it does not make a difference between the same word, capitalized or not (english Vs. English). In what concerns the training process, we developed it for 2 epochs, since we understood after several tests, that doing it for more than these epochs would result in overfitting.

IV. EVALUATION

a. RESULTS:

After understanding that using BERT would bring us more promising results, we wanted to still address the unbalanced dataset problem. To do so, we resorted to class weights. However, the results, that are exposed in the last 2 lines of Table 2, were not as favourable as expected, reason why we stuck to the original version, whose results are exposed in Table 3.

When comparing this model to our Baseline, analysed in the previous chapter, we augmented our Accuracy by 79,3%, from 0,29 to 0,52.

***Table 3** – Final Model considered for the Opinion Mining Project (BERT)*

Best Approach BERT			
	Precision	**Recall**	**F1-Score**
1: Anger	0,59	0,44	0,51
2: Anticipation	0,53	0,62	0,57
3: Disgust	0,55	0,53	0,54
4: Fear	0,49	0,38	0,43
5: Joy	0,46	0,55	0.50
6: Sadness	0,57	0,61	0,59
7: Surprise	0,44	0,47	0,45
8: Trust	0,46	0,43	0,44
Accuracy	-	-	**0,52**
Macro Avg	0,51	0,5	0,52

b. ERROR ANALYSIS:

We believe that our most common errors, as we can see in Table 3, is the inability to predict all emotions with the same (good) precision. Our model is better at predicting labels like "Anger" (in every 100 sentences that the model predicted with this label for the dev set, 59 actually had this label). However, the same is not true for some of the less represented labels, like "Surprise", which is the label with the lowest representation in the train data (only 8.13% of the sentences have this label).

V. FUTURE WORK:

Had we had more time, we would have clearly had chosen to focus on experimenting with another state of the art solutions of pre-trained models for NLP (in the same way we used BERT), such as *XLNet*, and tried to integrate them into deep learning algorithms, such as neural networks. On the other hand, we could have also tried *ELMo*, as embedding. In this way, the results obtained could have been slightly better. However, we believe that the results we were able to obtain throughout this project were quite satisfactory.

VI. ACKNOWLEDGEMENTS:

Also responsible for the successful completion of this project, we would like to thank and acknowledge our Text Mining Professor, Vânia Mendonça, for the available materials that were especially helpful on the part of data pre-processing and feature extraction; to Sandipan Dey, for its contribution on Stackoverflow (2022, March 27), and to the Language Technology at the University of Helsinki, for their contribution on GitHub (2022, May 19), that were both a precious help on using BERT with an unbalanced dataset, by using class weights; to *Thetechwriters*, from *AnalyticsVidhya* (December 20, 2021) for the BERT insights; to Pedregosa et al., from Scikit-learn: Machine Learning in Python (JMLR 12, pp. 2825-2830, 2011) for the valuable tutorials; and to *rashida048* from *regenerativetoday.com* (September 7, 2021), for the insights on models that were suitable for our baseline and intermediate approaches.

VII. REFERENCES:

Öhman, E., Pàmies, M., Kajava, K. and Tiedemann, J., 2020. XED: A Multilingual Dataset for Sentiment Analysis and Emotion Detection. In Proceedings of the 28th International Conference on Computational Linguistics (COLING 2020).

Lison, P., and Tiedemann, J., 2016. OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles. In Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016).

Jeff Reback, Wes McKinney, jbrockmendel, Joris Van den Bossche, Tom Augspurger, Phillip Cloud, gfyoung, Sinhrks, Adam Klein, Matthew Roeschke, Simon Hawkins, Jeff Tratner, Chang She, William Ayd, Terji Petersen, Marc Garcia, Jeremy Schendel, Andy Hayden, MomIsBestFriend, … Mortada Mehyar. (2020). pandas-dev/pandas: Pandas 1.0.3 (v1.0.3). Zenodo. https://doi.org/10.5281/zenodo.3715232.

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2. (Publisher link).

J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

da Costa-Luis, (2019). tqdm: A Fast, Extensible Progress Meter for Python and CLI. Journal of Open Source Software, 4(37), 1277, https://doi.org/10.21105/joss.01277.

Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc."

Richardson, L. (2007). Beautiful soup documentation. April.

Van Rossum, G. (2020). The Python Library Reference, release 3.8.2. Python Software Foundation.

N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of artificial intelligence research, 321-357, 2002.

TensorFlow Developers. (2022). TensorFlow (v2.8.2). Zenodo. https://doi.org/10.5281/zenodo.6574269.

Transformers: State-of-the-Art Natural Language Processing (Wolf et al., EMNLP 2020).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018, October 11). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv.org. https://arxiv.org/abs/1810.04805

Dey, S. (2022, March 27). Compute class weight function issue in "sklearn" library when used in "Keras" classification (Python 3.8, only in VS code). Stack Overflow. https://stackoverflow.com/questions/69783897/compute-class-weight-function-issue-in-sklearn-library-when-used-in-keras-cl

Huang, B. (2020, January 7). Text Classification with XLNet in Action. Medium. https://medium.com/@yingbiao/text-classification-with-xlnet-in-action-869029246f7e

https://www.facebook.com/jason.brownlee.39. (2019, March 12). A Gentle Introduction to the Bag-of-Words Model. Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-bag-words-model/

Joshi, P. (2019, March 10). What is ELMo | ELMo For text Classification in Python. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-features-from-text/

Language Technology at the University of Helsinki. (2022, May 19). XED. GitHub. https://github.com/Helsinki-NLP/XED/blob/master/BERT_COLING_NER.ipynb

Rani Horev. (2018, November 10). BERT Explained: State of the art language model for NLP. Medium; Towards Data Science. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

rashida048. (2021, July 9). A Complete Sentiment Analysis Project Using Python's Scikit-Learn – Regenerative. Regenerative. https://regenerativetoday.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn/

scikit-learn. (2019). Working With Text Data — scikit-learn 0.21.2 documentation. Scikit-Learn.org. https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

Scott, W. (2019, May 21). TF-IDF for Document Ranking from scratch in python on real world dataset. Medium. https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

Stanford NLP Group. (2009). Stemming and lemmatization. Stanford.edu. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

Thetechwriters. (2021, December 20). Multiclass Classification Using Transformers for Beginners. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/12/multiclass-classification-using-transformers/