**Project 3. Kaggle Text to Emotion Dataset 3**

Consider the dataset in <https://www.kaggle.com/datasets/praveengovi/emotions-dataset-for-nlp> where more than 600K records of Twitter dataset are classified into 8 categories: anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.

1. *Initial exploration*: You may inspire from some existing programs in Kaggle handling the same dataset, write a script that outputs the histogram of the different categories. Comment on the distribution of the training samples across various categories.
2. Write a script that concatenates all tweets of the same category and put them on a single file (data frame). Perform LDA with a number of topics equal to three and four keywords per topic. Write a script that performs this operation and save the result in a text file for each dataframe.
3. Now we want to test the cohesion of the records of dataframe by quantifying the extent to which the generated keywords are close to each others. For this purpose, write a script that evaluates the closeness of the three topics. To this end, we consider two metrics: a) I1: the overlapping between keywords of the three topics in terms of number of common words among all pairs; b) I2: highest semantic similarity between all pairs of the three topics calculated as the cosine similarity of the corresponding embedding vectors, where the embedding vector of a given topic is calculated as the average word2vec embeddings of the three keywords constituting the topic. Repeat this process for every dataframe. Summarize in a table the result for each dataframe in terms of I1 and I2 score.
4. We want to test the linguistic quality of the text in each dataframe. For this purpose, write a script that determines the number of tokens that correspond to stopwords and those that do not have entry in WordNet lexical database, as well as symbols, links and numerals. Then the linguistic quality ratio is provided as the ratio of the resulting cleaned tokens over the total number of tokens.
5. Use part-of-speech tagger to provide the number of verbs, nouns, adjective sand adverbs in each dataframe. Present the result in a table to summarize the result for every dataframe. Draw wordCloud of cleaned tokens in each dataframe. You may inspire from existing implementations available in Kaggle link (see also <https://github.com/amueller/word_cloud>). Discuss the consistency of the content data in each dataframe with the title of the category.
6. We want to test, the presence of negation in each dataframe. Write a script that detects the basic negation forms from the text, consisting of tokens: no, not, neither, never, no one, nobody, none, nor, nothing, nowhere. And Prefixes: un-, im-, in-, il-, and ir-, and dis-; As well as Sufixes: -less. Run the program to estimate the average number of negation terms occurring per record (one Tweet) in each category.
7. Use the provided splitting for training and testing dataset, and suggest a script that uses n-gram (n=2 and n=3) character TF-IDF feature with maximum feature vector size of 500, and SVM classifier to evaluate the F1-score in identification of every emotion category.
8. Repeat 6) when bag-of-word TF-IDF feature vector (size 500) was used. Test other alternative classifiers of your choice.
9. Study the state-of-the-art DeepMoji emotion recognition implementation in <https://github.com/huggingface/torchMoji> And use the package to calculate the F1-score on the testing dataset. If the time permit, you may seek to retrain DeepMoji with the training dataset available in this project.
10. We want to the quality of data augmentation on classification performance. For this purpose, consider the classes with small sample size (categories Disgust and Anticipation), identify from Kaggle or elsewhere labeled dataset of the same categories And add it to these two categories. Test the SVM classifier in 6) and discuss whether the overall performance can be increased or not.
11. From your previous investigation, feel free to do your own search and suggest an extra state-of-the-art approach for matching category content to category title. Identify appropriate literature to comment on the findings.