Term Frequency - Inverse Document Frequency

When working with textual data, there are several ways to obtain quantitative features that represent this data. As a reminder, each line of the dataset corresponds to an email and the functionalities will be different depending on the method of textual representation.

"Term Frequency - Inverse Document Frequency" (TF-IDF) is a score which represents the relative importance of a word in a tweet and in all the tweets of a classification. The TF-IDF reduces the impact of words that recur in a tweet but are empirically less informative than other words that occur less frequently.

$$TF(word, tweet) = \frac{frequency\ of\ the\ word\ in\ the\ tweet}{number\ of\ words\ in\ the\ tweet}$$

$$IDF(word) = \log \frac{number\ of\ tweets\ in\ the\ class}{number\ of\ tweets\ in\ the\ class\ where\ the\ word\ appears}$$

$$TF\ IDF(word, tweet) = TF(word, tweet)\ \text{x}\ IDF(word)$$

The TF-IDF matrix method is used in this part to represent emails.
Indeed, this method is known to obtain good results in the field of natural language processing, also the creation of features associated with an email is fast and finally parameters can be adjusted in order to avoid overfitting. The `TfidfVectorizer` method is therefore imported via the `sklearn.feature_extraction.text` package.

The parameters of the model used are presented in the following table:

Parameters	Values
`ngram_range`	(1,2)
`min_df`	4
`max_df`	0.7

An n-gram is a sequence of n words. Language models based on n-grams define the conditional probability of the n-th word given the previous n-1 words. The model uses the products of these conditional distributions to define the probability distribution over sequences of words:

$$P(x_1, \dots, x_\tau) = P(x_1, \dots, x_{n-1}) \prod_{t=n}^{\tau} P(x_t | x_{t-n+1}, \dots, x_{t-1})$$

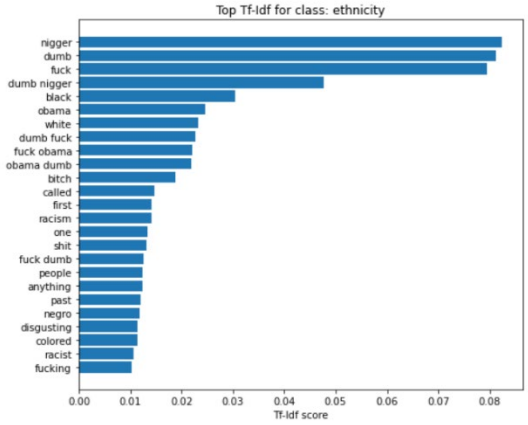
Training an n-gram model is straightforward because the maximum likelihood estimation can be calculated by counting how many times each possible n-gram occurs in the training set. For n = 2, the model is called a bi-gram. Thus, a bi-gram should help improve the performance of the model by taking into consideration the words that are likely to appear together in emails.

When building the vocabulary of the model, the Minimum Document Frequency (resp. Maximum Document Frequency) makes it possible to ignore the words which have a frequency, in all the emails, lower (resp. higher) than the indicated threshold:

- `max_df` = 0.7: remove words that appear in more than 70% of the tweets in the class.
- `min_df` = 4: remove words that appear in less than 4 tweets in the class.

The next step is to use the `TfidfVectorizer` on *(name of the column)* of the `X_train` and the `X_test`. This step makes it possible to extract the features of *(name of the column)* and to return a parsimonious matrix with the TF-IDF scores of each word of the vocabulary present in all the tweets.

Thus, the training and test sets matrices used during the modeling are parsimonious matrices containing around 18,000 features. For each class (or types of cyberbullying), a list of the 25 words or sequences (bi-grams) that are the most important in identifying the class, according to the average of their TF-IDF score for all the documents of this class, is created.



For the type of cyberbullying: ethnicity, the words *nigger*, *dumb* and *fuck* are the most important in predicting the class. Their average score of TF-IDF on all the tweets as ethnicity, are indeed the highest. These results seem coherent.

Nevertheless, results for the classes not cyberbullying and other cyberbullying have lots of similarities and there are not representative words that seem to clearly predict these two classes. As so, we expect to see many misclassifications between these two classes.