**Quora Insincere Questions Classification**

**Methodology:**

First of all, we have used the Pandas library from Python to read and prepare the data of the files *“train.csv”* and *“test.csv”.* We have divided the train dataset into two subsets, one for training and one for validation. To make this division we have used the ‘model\_selection’ function from ‘sklearn’ with a validation size of 20%. The validation set is the one we use to check the performance of the methods that we are proposing.

We have mainly used three different ways to encode the document: *CountVectorizer*, *TfidfVectorizer* and *word2vec*. They are already implemented in some libraries of Python, so we have just been modifying them and changing their hyperparameters.

For *CountVectorizer* and *TfidfVectorizer* we have been trying the following alternatives for the feature vector creation:

* **Lemmatization**: we have used the *WordNetLemmatizer*
* **Steaming**: we have used the *PorterStemmer*
* No additional preprocessing

In all these three cases, we have been trying as analyser different *N-grams* ratios. In addition, we have tried to train the models using and not using *stop-words*. We have been trying as well some preprocessing by hand. We will see in the section of Experiments the results obtained.

For the *word2vec* experiments, we have implemented the ‘*word2vec’* function from the *‘gensim.models’* library. In general, in these experiments we have used a very simple preprocessing of the data: just splitting and converting the text to lowercase. One reason of this procedure is that we have seen that there are some questions in the dataset written in other languages such as Russian, Arabian or Chinese. So, if the *word2vec* does not have these kinds of words in the construction of the vocabulary, when we use the regressor (i.e. the Logistic Regressor) we are going to obtain a lot of NaN values.

After having the feature vectors, we have used the function from *sklearn* to create a Pipeline with the vectorizer, the feature selector and the classifier used. Then we have done Cross Validation using a randomized search for the model selection instead of grid search, a bit different depending on the classifier used (different hyperparameters). We have chosen randomized search because the results in parameter settings is quite similar, while the run time for randomized search is drastically lower. We have then used the functions *.fit* and *.predict* to train the model and get the accuracy and f1 score.

The thing that we commonly do in all the methods is the features selection. We use the function from *sklearn* *SelectKbest*, taking as score function chi-squared. We select the value of k, the number of top features to select, doing Cross Validation over the range between the minimum and maximum number of features available.

The classifiers we have evaluated using the different procedures explained before are:

* Logistic Regression
* XGBoost
* Multinomial Naïve Bayes
* LSTM

For these classifiers, we have done Cross Validation in different hyperparameters:

* Logistic Regression: Inverse of regularization strength C
* XGBoost: Learning rate
* Multinomial Naïve Bayes: Additive smoothing parameter alpha

In the notebook you can see how we have implemented all the previously explained.

**Experiments:**

In this section we present the results obtained with all the different methodologies used, that we have explained previously. We have done cross validation for the hyperparameters, but still has been quite a manual process to choose which classifier or vectorizer to use, as well as the parameters of it. The following table present these results: