IML Hackathon 2020

Group Members: Yonatan Chamudot 312516289, Ben Schreiber 88880063, Moshe Tannenbaum 333854032, Jason Greenspan 336126362

June 10-12, 2020

Task Chosen: Github repositories classification

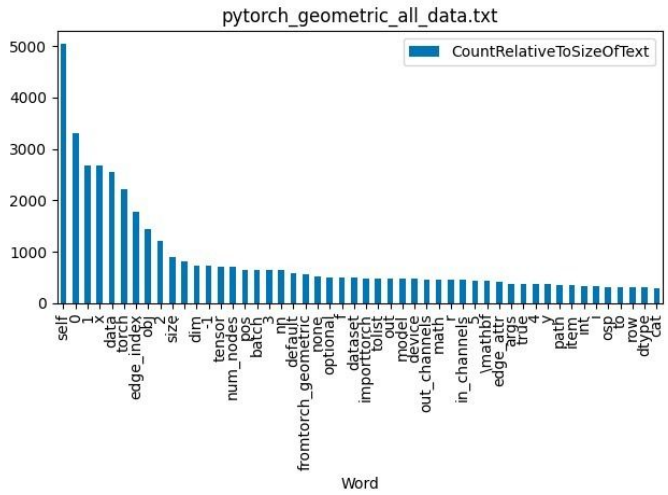
Description:

We initially did not have a strong instinct for how to approach a code-classification problem. The challenge was two-fold: firstly we had no hands on experience with multi-class classification, and secondly we did not know which features we could extract from snippets of text that would be useful in training a classifier.

We started out with trying to write a base-line classifier that would take snippets of code and output a vector of scores along 2 features: the amount of whitespace characters as a percentage of the length of the snippet, and the amount of single and double quotation mark characters. We thought that these two features may give a rough indication of the coding styles used in each repository, and put code in a features.py module for grading each snippet along the feature-axes. We chose to use a One vs One multiclass classifier from sklearn's multiclass package. Based on online research, we found that the best base algorithm for text classification is SVM since SVMs are able to handle high-dimension feature vectors well.

After further research, we found a paper at https://clgiles.ist.psu.edu/papers/KDD-2002-whatsthecode.pdf that described a solution to a similar problem, based on comparing the snippets to a list of the most common words or lexical tokens found in the source files. With this new approach in mind, we wrote a MostCommonWords module that found the 50 most common

words in each repository, and combined them into a master list of 350 words. We filtered out words that were common in all repositories, such as '='. We added a score_vocab function that takes the data set and outputs a design matrix of 350 features, where a 1 would indicate that the word was present in the snippet and a 0 would indicate that it was absent.



Graph of the most common words appearing in one of the repositories

Due to runtime considerations, we decided to train the classifier on (at most) 5000 lines of code from each repository. Once we were able to train the classifier without errors, we serialized the classifier object and saved it in an external file. We then tested the classifier's predict function on 5000 test strings from the original files. In our tests we were able to achieve an accuracy of about 54%. After randomizing the selection of lines for the test set, we achieved an accuracy rate of about 62%, more than 4 times what would be expected with random classification.