**NLP: Text Categorization Report**

**System Overview**

The program was written in Python 3.9 and developed on a macOS operating system. Prior to running the program, the necessary libraries can be installed by downloading them from the requirements.txt file, which is done by typing the following command in the terminal of the directory containing the file: **pip install -r requirements.txt** It should be noted that Python and pip package installer must be installed on the machine prior to downloading the requirements and running the program. In order to run the program, assuming a set of labeled test documents exists, the user can run the script, text\_categorization.py, by typing the following command in the terminal of the directory containing the script and the necessary input files: **python3 text\_categorization.py**

If there is no set of labeled test documents available, the user can run the script, split\_data.py, to randomly split a set of labeled training documents into a smaller training set, labeled validation set, and unlabeled validation set using a chosen test size by typing the following command in the terminal of the directory containing the script and the labeled training documents file: **python3 split\_data.py**

The source code and other necessary files for the project can be found on [GitHub](https://github.com/chrislee8684/NLP_text_categorization).

**KNN Classifier**

The machine learning method used for text categorization was a KNN classifier. The program tokenizes the training and test document texts by using the word\_tokenize() function of the nltk library. To be more specific, the “punkt” tokenizer, which is a pre-trained model, is downloaded and used to tokenize the texts. The TF\*IDF weighting scheme was used for the tokens. An implementation of the stop list that is provided with the nltk library was attempted in order to remove trivial words from the vocabulary, but the accuracy did not change much (decreased in some cases), so it was removed.

For calculating the similarity scores, both Euclidean distance and cosine similarity methods were attempted. It should be noted that utilizing the vectors as NumPy arrays decreased the computation time by a significant amount. The results are shown and discussed in the section below.

**Results**

In order to compare the methods of Euclidean distance and cosine similarity, an arbitrary value of K=5 was chosen and both methods were evaluated for each corpus. As shown below, the cos similarity resulted in significantly higher accuracies across the board, so it was chosen as the optimal method. Arbitrarily low and high K values of 1 and 20 were attempted with cos similarity for Corpus 1, but the resulting accuracies were similar, which suggested that K values are insignificant for accuracy compared to the significance of the computational methods.

Corpus 1:

K = 5, Method = Euclidean distance

A screenshot of a computer

Description automatically generated

K = 5, Method = Cos similarity

A screenshot of a computer

Description automatically generated

K = 1, Method = Cos similarity

A screenshot of a computer

Description automatically generated

K = 20, Method = Cos similarity

A screenshot of a computer

Description automatically generated

Corpus 2:

For corpus 2, a 0.2 test size was chosen and the following files were created for evaluation: corpus2\_train\_subset.labels, corpus2\_validation.list, corpus2\_validation.labels

K = 5, Method = Euclidean distance

A screenshot of a computer

Description automatically generated

K = 5, Method = Cos similarity

A screenshot of a computer

Description automatically generated

Corpus 3:

For corpus 3, a 0.2 test size was chosen and the following files were created for evaluation: corpus3\_train\_subset.labels, corpus3\_validation.list, corpus3\_validation.labels

K = 5, Method = Euclidean distance

A screenshot of a computer program

Description automatically generated

K = 5, Method = Cos similarity

A screenshot of a computer

Description automatically generated

**Future work & Recommendations**

To determine an optimal K value, either cross validation methods or a for loop of various K values can be evaluated and the resulting accuracy values can be compared.