Project 1: Text Categorization

Leart Krasniqi
Prof. Sable
Spring 2020

1 Overview

The goal of this project was to create a text categorization system. The system uses a set of training documents to learn word statistics and then applies those statistics to test documents in order to categorize them. The system consists of two programs:

- `train.py`, which prompts the user for a set of training documents and then saves word statistics to a JSON file.
- `categorize.py`, which prompts the user for a set of testing documents and word statistics, and then categorizes the test documents.

Both programs utilize the Natural Language Toolkit (NLTK) and the `progress` package (in order to display a progress bar) from Python.

2 Naive Bayes Classifier

The system utilizes a Naive Bayes classifier. In both the training and the testing programs, the built-in NLTK `word_tokenize()` function is used. Initially, this was the only method used for tokenization, but it was found that applying stemming (via the NLTK `PorterStemmer()`) generally improved accuracy when used in both the training and testing programs. The classifier employed Laplace smoothing, with a k-value of 0.055. Section 3.2 explains why this value was chosen.

3 Performance Evaluation

3.1 Training and Testing

In order to test the categorization system on Corpora 2 and 3, the training sets for each were split into a training set and a tuning set. A separate Python script was written to split an original training set into x% training and $(100 - x)$% tuning, where x is a command-line parameter that the user can supply. The splitting is done by using x% of each category's files for training and the remaining for testing. For each of the provided corpora (Corpus 2 and Corpus 3), 70% of the original training set was used for training and the remaining 30% was used for testing.

3.2 Smoothing Selection

Originally, simple add-one smoothing was used in the Naive Bayes classifier, which yielded decent results. In order to find the optimal smoothing variable, another Python script was written such

that the categorization system is run with multiple k-values and their accuracies were compared. Figure 1 shows the results of running the script:

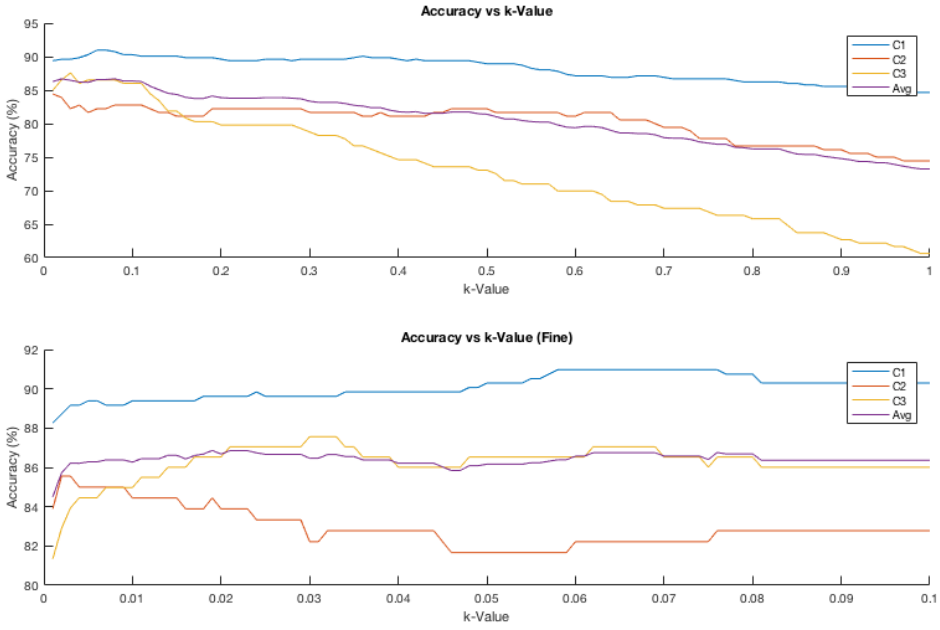

Figure 1: Accuracy of System at Different Smoothing k-Values

Initially, the system was tested at 100 k-values ranging from 0.01 to 1.00 (Top graph in Figure 1), but it was found that the system performed best at very small k-values. Therefore, the system was tested once more at 100 k-values now ranging from 0.01 to 0.1 (Bottom graph in Figure 1). Although a k-value of 0.055 did not maximize the average accuracy, a compromise was made to keep the accuracy of Corpus 1 (that had a full training set) high, while also keeping the average accuracy high.