Jason Kurian
NLP Project 1 – Write-up

Instructions to Run Program

In the provided zip are three files:

1. text_categorizer.py - the main program to be run, which prompts for the three files
2. rocchio.py – module imported into text_categorizer.py
3. run.sh – a Bash script I used to run text_categorizer.py on the three datasets (assuming a uniform naming scheme) and call the Perl script

Note that the program needs to be in the same directory as the files it receives as input, along with the directories containing the corpora. I ran this program with Python 3.8 and installed all NLTK libraries by running *"import nltk; nltk.download()"* in the Python terminal.

Project Overview

Machine learning method: Rocchio/TFIDF

Tokenization

Before tokenizing the files, I convert all the file contents to lowercase and remove any numbers and punctuation. I then use the Regexp Tokenizer provided by the NLTK package to convert the file content to unigrams. I then filter out any tokens that are in the NLTL stop words list and, after using the NLTK POS tagger, are not either a noun or verb. I use the WordNet Lemmatizer to lemmatize the tokens, before lastly appending the POS abbreviation to each token.

TFIDF Calculation

Instead of computing the generic TFIDF values for each document, I modify the term frequencies to match how they are calculated in BM25 as follows:

$$TF = TF/(TF + k * \left(1 - b + b * \frac{dl}{adl}\right))$$[1]

where *k* is a tuning parameter to control the effect of term saturation, *dl* is the document's length, *adl* is the average document length across the training corpus, and *b* is a tuning parameter to control the effect of document length. This modification ensures that the final TFIDF scores account for term saturation (so that increasing occurrences of a term does not linearly correspond to increasing relevance) and document length. The parameters *k* and *b* were chosen manually by iteratively testing over different combinations and choosing the one that resulted in the best accuracy across all datasets.

Testing

I evaluated the performance of the text categorizer on the second and third datasets by randomly setting two-thirds of the files as the training set and the remaining third as the test set. I used a script to test all the datasets at once to determine the optimal configuration of features and hyperparameters. Accuracy improved when using unigrams over bigrams and trigrams, using solely lemmatization over stemming or a combination of the two, and using the relatively small NLTK stop words list over larger stop words lists I found online. The BM25 calculation for term frequencies provided a marginal benefit to accuracy (additional 1-2 correct files), while using only nouns and verbs and appending the tag to the tokens significantly increased the accuracy.

[1] http://www.kmwllc.com/index.php/2020/03/20/understanding-tf-idf-and-bm25/