Lab 1

James Lowrey

In this lab two different methods were used to parse Reuter’s articles in order to find “interesting” or meaningful words. The Reuter articles are encoded in a sgm format with tags identifying the topics, places, title, and body of each article. Words are interesting/meaningful if they indicate information about the article they are in. Stop words like “and”, “a”, or “the” do not indicate information as they are very common in the English language. Extremely rare words also do not indicate information, as they can be noise in the text extraction process. Thus different processes were employeed to find words that match this middle of the line criteria.

The first step in this information retrieval task was finding the relevant text in the sgm files. The BeautifulSoup4 Python library was utilized to turn the sgm files into a navigable tree. The BeautifulSoup4 library lacks a sgm parser, so an html parser was used instead. This ended up being a beneficial design choice, as html parsers are generally more lenient than xml or sgm parsers when it comes to poorly formatted articles. The given data files had some poorly formatted tags (not closed properly, slightly mis-spellinged) which BeautifulSoup4 took care of in its parsing.

After the tree is created it can be traversed using BeautifulSoup4’s API. This tree contains a series of Reuter nodes, and each Reuter node may contain topics, places, and title nodes, as well as an associated text (representing the body text of the article). From this tree a class label and a count of words from the article’s text are created. The class label is composed of the topics, places and title information. After the class label, the words from the body text are counted. A first round of word filtering is applied when counting.

To count the words the Natural Language Toolkit (NLTK) python library is first used to break the article text into word tokens. Then each word token has its punctuation removed. Finally, the first round of word filtering occurs by removing numbers, small tokens, and stop words. Numbers are removed as they are meaningless without context. The same applies for dates (ex: “9/12/15”), as they will be treated as a number with punctuation removed. Small tokens are defined as a word token with length less than two characters. This would delete any token that was previously all punctuation (ex: “...”) or mostly punctuation (ex: “-a-”). Stop words are words that do not mean anything on their own but are required for grammatically correct sentences.

After filtering out the special cases, the word is stemmed and counted. Stemming is a process that attempts to map common variants of words down to their root word. While counting, every unique, unfiltered word is saved to a set (the first row of the eventual data matrix) in order to keep track of how many unique words are in the corpus. Before this initial filtering there are over 60k unique word tokens in the corpus. Afterwords there are about 43k unique tokens.

During the first round of filtering/counting, the number of articles a word occurs in is also counted and saved for later. These counts are important for document frequency filtering and Term Frequency-Inverse Document Frequency (TF-IDF) filtering. Document frequency filtering removes terms that appear in very many or very few documents. For this lab, terms were removed if they occurred in >99% or <1% of the articles. These numbers were determined experimentally to ensure the final data matrix had approximately 1,000 unique words.

TF-IDF filtering is a practice that leverages the number of times a word/term/token appears in its own document against the number of times that term appears in all other documents in the corpus. This value reflects how important a term is to its document, offset by the frequency of the term in the corpus. TF-IDF is performed on every term after the first round of filtering, and the top 500 terms are output as the TF-IDF feature vector. These 500 terms are created by finding the top 15 TF-IDF terms in every document, and then searching for the top 500 amongst the result.

Finally the feature vectors are printed to the console. The first vectors consists of the top 500 TF-IDF terms in the corpus. The second is a type of data matrix, where the first row is a set of all unique words in the filtered corpus and all subsequent rows are lists representative of a Reuter's article. The first element is the Reuter's class label and the second is a count of interesting words in the Reuter's article's body.

One filtering method that was implemented and later removed was filtering out terms based upon overall count in the corpus. The total number of times a word token appeared in the corpus was tracked and later removed if it was above or below certain thresholds. This method was removed in favor of document frequency filtering as the later method more accurately tracks the “common-ness” or “interesting-ness” of a given word.