Siddhanth Satish

COMPSCI 446 TOKENIZATION PROJECT REPORT

Location of Part A and Part B in the CODE

Part A and Part B have been implemented between lines 126- and 147- with the help of helper functions that aid the tokenizing, stop-word removal and stemming process. Their results have been written into tokenized.txt and terms.text respectively and attached with this submission. Part B implementation also generates a graph using matplotlib as requested in question 6.

Description of System:

- Both Part A and Part B first open and read the input text file, storing their data as a string variable.
- Next, tokenization, stopword removal and porter stemming is performed on them (details on each below)
- Finally their results are written onto a text file. (Part A- tokenized.txt, Part B- terms.txt)
- Part B has an additional method, vocab_graph() which records the growth in vocabulary. (details below)

1. *tokenization()*

- Fixing abbreviations with re.sub. Regex used: `'(?<!\d)\.(?!\d)'` checks if there is a . between any two elements which are not digits. This way I am not only covering abbreviations like U.S.A., but also checking for situations like pH.D. and making phd. Designed in such a way this works for both lowercase and uppercase. The method correctly reacts to digits by making 200.000 and 200,000 to [200,000]. It only fails when I have terms en.wikipedia.org as enwikipediaorg.
- Method handles contractions by simply removing all apostrophes in the document with the string replace() function. Similarly it makes all the letters lowercase with the lower() function.
- Finally I tokenize the document using the re.findall functions to treat all the punctuations I know of as word separators.

2. *stop_word_removal()*

- This is a simple method where I iterate through all the words in our tokenized document and remove the words that are present in the given stop-word list.

3. *porter_stemming()*

- In 1a and 1b we are removing parts of words to make processing them faster
- 1a examples: gaps-> gap, stress -> stress, stresses -> stress, misses -> miss, ties -> tie, cries-> cri
- 1b examples: agreed -> agree, feed -> feed, pirated -> pirate, sled -> sled, medly -> medly, supposedly -> suppose, pirating -> pirate, sing -> sing, falling -> fall, dripping -> drip, hoping -> hope

4. *vocab_graph()*

- Here we are iterating through the list words keeping track of its count and the frequency in which new words occur. Then we are graphing this change in frequency using matplotlib.

Siddhanth Satish

Software Libraries Used:

Python libraries: re, collections, matplotlib

- Re: For regular expression tasks.
- Collections: For using Counter to count the most common 300 words.
- Matplotlib: For graphing the frequency at which new words occur.

Two Changes in Tokenization and Stemming:

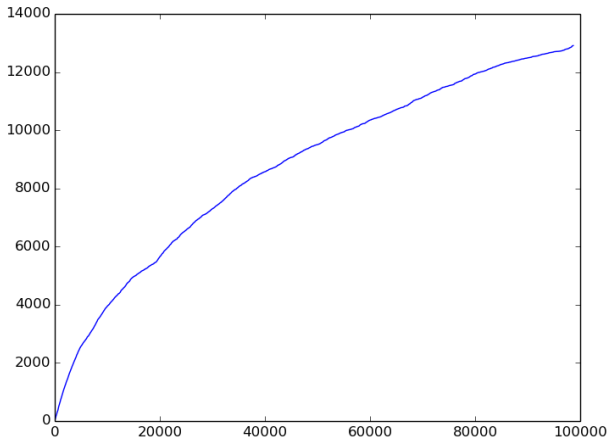
1. ***Tokenization***
   - ➢ Change in a way that phrases like en.wikipedia.org get recognized as [en, wikipedia, org]. Now they are tokenized as [enwikipediaorg]
   - ➢ Change in a way that phrases like t-mobile and e-bay get recognized as [tmobile, ebay] Now they are tokenized as [t, mobile] and [e, bay]
2. ***Stemming:***
   - ➢ Change in a way that verbs like doing and going become do and go. Rule 1b according to my implementation makes it doe and goe.
   - ➢ Develop a method/ algorithm to extract stems out of misspelled words

Most Frequent Words

Top 10 most frequent words: {whale, ship, man, ahab, sea, boat, old, head, time, look}

- Yes, the words whale, ship, sea, ahab, head and boat are very relevant to the story.
- Whale is moby dick, ahab is a character, ship, sea and boat are recurring places in the story.
- Some words like head, look, old, and man are words we use everyday.
- They could even be used as stop words when the corpus of words is very large.

Moby Dick Vocabulary Growth Graph



X Axis: Total Number of Words in Document

Y Axis: Total Number of Words in Vocabulary