HW Assignment 1 (Due date: Fri, Oct 4, by 10:30am)

1 Text Processing [100 points]

CACM is a collection of abstracts of articles published in the Communications of the ACM journal between 1958 and 1979. This collection has been used in numerous papers for the evaluation of information retrieval systems. The entire collection (3024 documents) is available from the class web site and is comprised of 3 files:

- *cacm.tar.gz* contains the 3024 html files.
- *cacm.query* contains the 64 queries.
- *cacm.rel* contains the relevance query results for each of the 64 queries.

Write code that eliminates all HTML tags and then tokenizes the text. Ignore the numeric columns at the end of each file. Use the tokenized text to answer the following questions:

1. How many tokens are in the entire collection?
2. What is the size of the vocabulary?
3. How many vocabulary entries are *happax legomena*? How about *dis legomena*?
4. What are the top 20 most common token types?
5. What is the percentage of tokens in the collection that is covered by these 20 most common token types?

Further process the text by eliminating stopwords and do stemming using Porter's algorithm. Use the stemmed version of the text to answer the same questions.

Use the tokenized version of the text to answer the following word distribution questions:

6. Compute the total number of tokens and the vocabulary size for the first half of the collection (the first 1512 documents). Use these numbers, together with the numbers from points 1 and 2 above to derive the parameters k and b for Heap's Law.
7. Estimate the size of the vocabulary if the collection grows to 1 million tokens.
8. Plot the frequency (f) vs. rank (r) graph for all tokens in the vocabulary, similar to the graph on slide 4, lecture 3. Also plot the $log(f)$ vs. $log(r)$ graph.
9. (*) Use linear regression to find the constants c and k such that $log(f) = c + klog(r)$ best approximates the graph above. Plot the resulting line together with the previous graph.

Relevant functionality in NLTK: functions `nltk.clean_html`, `word_tokenize` and `sent_tokenize` from package `nltk.tokenize`; class `PorterStemmer` from package `nltk.stem.porter`; the `stopwords` corpus from package `nltk.corpus`.

2 Word Clouds [100 points]

The Wordle applet at `www.wordle.net` is a toy for generating *word clouds* from text that you provide. The clouds give greater prominence to words that appear more frequently in the source text. On the class website you can find a link to a word cloud that has been generated from a text file that contains the descriptions of all computer science courses taught in the School of EECS.

1. Looking at the word cloud, we can see that singular nouns (e.g. *communication*) and plural nouns (e.g. *communications*) are represented separately, even though they refer to the same concept. Write a function that reduces all plural nouns to their singular form, and then generate the corresponding word cloud.

2. Generalize the function above to reduce all tokens that have the same stem to the same word form, and then use the resulting word forms to generate a word cloud. For example, *program*, *programming*, and *programs* have the same stem, therefore they can all be reduced to the same word form, i.e. *program* (the shortest one).

You are encouraged to propose alternative processings of the text. The best word cloud(s) will be used by our ACM student chapter for a T-shirt. [*Based on a CS word cloud idea by Dr. David Juedes*]

Relevant functionality in NLTK: function `pos_tag` in package `nltk.tag`, class `WordNetLemmatizer` from package `nltk.stem.wordnet`.

3 Inverted Index [100 points]

Create an *inverted index* for the CACM collection, using the output of the text processing done for problem 1. The index will consist of the following files:

1. A file that maps each term name to: the term ID, the document frequency, the offset of the posting list in the index file, and any associated term information you think might be useful. Store one mapping per line.

2. A file that maps each document ID to: the document name, the document length, and any document associated information that you think might be useful. Store one mapping per line.

3. The inverted index file that maps term IDs to their postings lists. Each posting should contain a document ID and the term frequency in that document. Store one postings list per line.

The inverted index file constitutes the bulk of the index. For simplicity, you can build this file in stages:

1. Maintain a separate file per unique term, adding document information to these files as you process the documents.

2. Concatenate these files into one inverted index file and add the appropriate inverted index offset to the term file.

Write code that takes as input a pre-processed text collection such as CACM and creates the index files above. Write a function that takes as input a term and an inverted index and outputs the term information and its postings list.

4 Submission

Please turn in a hard copy of your homework report at the beginning of class on the due date. Electronically submit a directory that has your working code, data, and images, and a concise README file describing these before class. Do not send NLTK code! Create a gzipped, tar ball archive of your directory, and upload it on Blackboard.

For example, if the name is John Williams, creating the archive can be done using the following commands:

> tar cvf williams_john.tar williams_john

> gzip williams_john.tar

These two steps will create the file 'williams_john.tar.gz' that you can upload on Blackboard.

Please observe the following when handing in homework:

1. Structure, indent, and format your code well.

2. Use adequate comments, both block and in-line to document your code.

3. Type and nicely format the project report, including discussion points, tables, graphs etc. so that it is presentable and easy to read.

4. Working code and/or correct answers is only one part of the assignment. The project report, including discussion of the specific issues which the assignment asks about, is also a very important part of the assignment. Take the time and space to make an adequate and clear project report.