

Homework #1 (due in 2 weeks)

Administrata (5 points)

In the next week, send me an email at paulmac@apl.jhu.edu identifying yourself and telling me which email address you would like class-related mailings sent to. Any account is fine, but it should be one you can check regularly.

Corpus Statistics (75 points)

Zipf's law (cf. Section 5.1 in the text) predicts that the number of times the i th most frequent word will be seen is about k/i times the frequency of the most common word, for *some* k . You can verify whether this is so by examining a collection of text and counting the number of occurrences for each word. For this purpose, an electronic text of declassified CIA reports has been placed on the course web page. For this assignment, download this file, and then write a program (or programs) that:

- Performs some normalization of the text. For example, removing punctuation and repairing some errors from the text.
- Reports the number of 'paragraphs' processed, the number of distinct words observed, and the total number of words encountered. Be sure to describe how you determine what constitutes a word. For example, you should indicate if you ignore case-differences and how you treat punctuation symbols and digits.
- Stores the total number of times each word is seen and the number of documents the word occurs in.
- Identifies the 20 most frequent words (by total count) and reports the number of times each occurs (both in total and expressed as the number of documents containing it).
- Also prints the 100th, 500th, and 1000th most-frequent word and their frequencies of occurrence. (Don't submit a printout with the top 1000.)
- Computes and prints the *number* of unique words, which occur in exactly one document. (I believe Chicago is one such word.) What percentage of the dictionary terms occur in just one document?

You can (and should) build the required lexicon in one scan of the input text. After sorting terms by total frequency it should be fairly easy to extract the pieces of information I ask you to report. In the file paragraphs are indicated with <P ID=XXXX> tags indicating the start of each new paragraph. Some 'paragraphs' will be very short; some are empty.

Lexicons are a key data structure for IR systems. In future assignments you will need a lexicon, so you might consider how to make your dictionary modular and reusable. In particular you will want it to fit in memory, to be storable on disk for subsequent reloading, and to enable efficient lookups. Some representations you might consider are in-memory hashables, binary trees, or tries. If you use Java, using the built-in HashMap or a TreeMap classes in Java is a very reasonable idea. Hand in your source code and the requested output described above.

For this assignment the document text was created by downloading the CAESAR, POLO, and ESAU papers from <http://www.foia.cia.gov/cpe.asp> and converting PDF to text with the *pdftotext* tool. Each 'page' was placed in a single paragraph tag. The documents are scanned images of typewritten pages, which contain redacted sections, which explains why there are so many errors in the text. The collection covers a 20-year period from 1953 to 1973. The text is about 24 MB uncompressed.

Questions (5 points apiece)

- [1] Who coined the term 'information retrieval' and when? Cite your source and *briefly* say how you found the answer.
- [2] Problem 1.7 (pg. 13)
- [3] Problem 1.9 (pg. 13) [note: consider 'optimal' to mean using the least possible amount of processing effort]
- [4] Problem 2.8 (pg. 41)

For More Fun (extra credit, 5 points)

Amazon provides 'statistically improbable phrases' for books (e.g., for the course text some examples are 'ranked retrieval results' and 'wildcard queries'). Can you identify interesting phrases from the CIA collection? You might start by looking at the most frequent two, or three word phrases. How, if at all, does computing frequencies for phrases affect the time and memory requirements of your program compared to individual words?