Programming Assignment #1 (due in 1 week)

Corpus Statistics

Zipf's law (cf. Section 5.1 in the text) predicts that the number of times the *ith* most frequent word will be seen is about *k*/*i* times the frequency of the most common word, for *some k*. You can investigate whether this is so by examining a collection of text and counting the number of occurrences for each word. For this assignment, links to two electronic texts have been placed on the course web page: a collection of ~ 9k reviews from Yelp and 500k news headlines. Download these files and write a program that you will run separately on each text. Your program should:

- Perform some normalization of the text. For example, split on spaces, remove punctuation, and lower-case words. Give some thought to it. You may, but you are not expected or required to use a stemming algorithm. Do not perform stopword removal.
- Report the number of 'paragraphs' processed; we'll consider each paragraph to be a 'document', even though all paragraphs are contained in a single file[1].
- Report the number of unique words observed (*vocabulary size*), and the total number of words encountered (*collection size*, in words).
- Calculate both the total number of times each word is seen (*collection frequency* of the word) and the number of documents which the word occurs in (*document frequency* of the word).
- Identify the 100 most frequent words (by total count, also known as collection frequency) and report both the collection frequency and the document frequency for each. Order them from most frequent to least.
- Do the same for the 500th, 1,000th, and 5,000th most-frequent words. (But do **not** just printout the top 5,000.)
- Calculate and print the *number* of words that occur in exactly one document. I do not want to see the words themselves. For *Yelp,* I believe *lemongrassy*, *mercedes*, and *nepali* are such words. Also calculate the percentage of the dictionary terms that occur in just one document?

Run your program on both datasets, and in addition to the desired output described above, provide a short writeup that: (a) describes how you normalized the text and determined what a "word" is; and, (b) summarizes any similarities and differences in the top-100 terms from *Yelp* and *Headlines*.

You can (and should) create the required lexicon[2] in one pass over the input text. After sorting terms by frequency it should be easy to extract the pieces of information that I am asking you to report. In the input files paragraphs are indicated with <P ID=XXXX> tags indicating the start of each new paragraph. Some 'paragraphs' are short, some are longer; it may be that none are empty, but I have not verified this.

Lexicons are a key data structure for IR systems. In future assignments you will need a lexicon, so you might find it worthwhile 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 support efficient term lookups. Some representations you might consider are in-memory hashtables, binary trees, or tries. If you use Java, using the built-in HashMap or a TreeMap classes is a very reasonable idea. Provide your source code, program output, and writeup (as described above) in a single PDF file.

I have coded a solution to the exercise above in ~240 lines of (verbose, commented) Java code and in previous classes I have had former students submit good, readable solutions in about 2 pages of quality Perl[3] or Python. Of the five programming assignments, this is by far the most simple.

The Yelp data is a sample from the Yelp Academic Dataset. The headlines are from Sept. 2015 and came from the NewsIR'16 workshop (data from Signal Media). The files are about 7MB (Yelp) and 40 MB (Headlines). I made a good faith effort to remove obvious vulgarity from the Yelp data, but it is real data from the Internet.

If you have questions about the assignment you can post them to the discussion forum, or shoot me an email.

---

[1] This is a lot easier than for me to give you 20,000 separate files to work with.
[2] You are building a dictionary. The words dictionary and lexicon are interchangeable.
[3] I once lost a bet to a colleague who likes obfuscated programming contests. My Perl solution was one line and 338 bytes. His was smaller.

Below are some short samples from the texts:

From *Yelp:*

<P ID=1322>
I gave this place 5 stars because of their constant improvement against all the new Korean BBQ restaurants opening in Vegas. When this place first started, the menu was based very consistently around tofu dishes but now, they are more on the BBQ edge. I went there about 1 week ago and the galbi was marinated nicely, the taegi bulgogi was very fresh, good portions, and 10 dishes of panchan, very fresh lettuce, and the service was incredible. It was very reasonably priced as well. I'm definitely coming back as always.
</P>
...
<P ID=1332>
Really bad food, I was in Las Vegas craving for indian food and I visted this place... the food here is hopeless they serve frozen products as freshly prepared food... I think this place is a joke.... the food is really below par, I mean$13.00 for a veggie that too, awefully prepared.... Waiter was rude and unprofessional... served food as if he was doing us a favor
</P>

From *Headlines*:

<P ID=64811>
Johns Hopkins and DuPont Join Forces to Produce an Improved Ebola Protection Suit
</P>
...
<P ID=103002>
Mana Health Sponsors Johns Hopkins University's First-Ever Medical Hackathon, MedHacks, Oct. 2-4
</P>
...
<P ID=147755>
U.S. Congressman Elijah E. Cummings To Be The Keynote Speaker at BOLD Conference at Johns Hopkins Applied Physics Laboratory In Columbia, Maryland
</P>

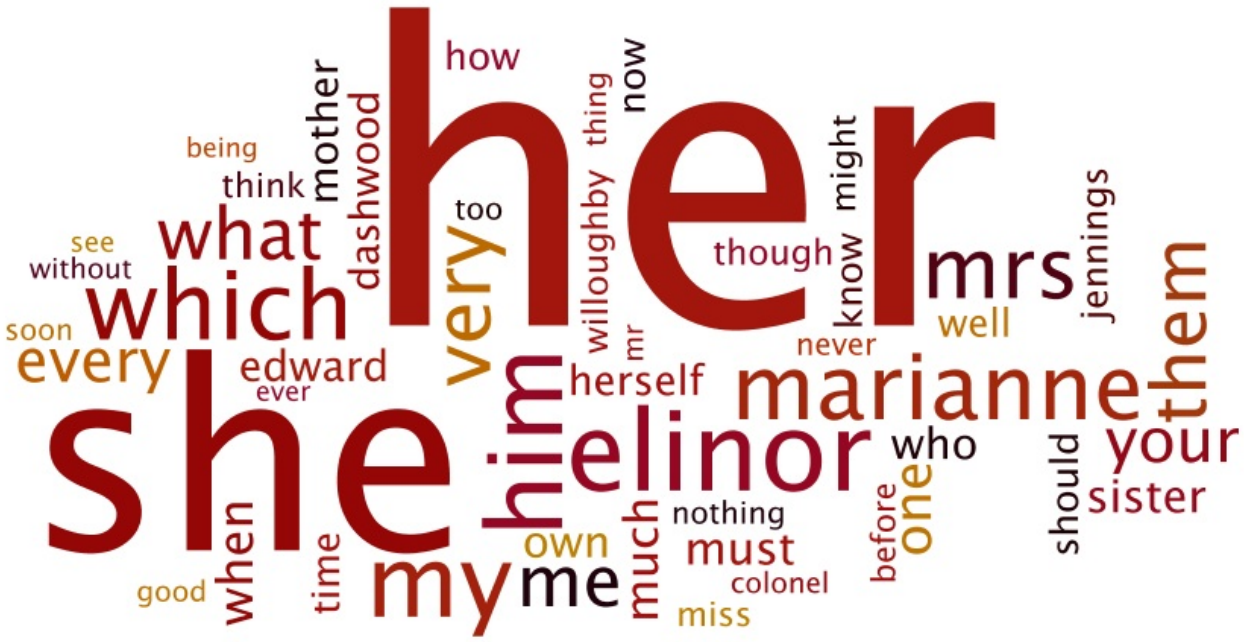
Note the “P” tags are on separate lines from the text. Other datasets used in later assignments will be formatted similarly.

For More Fun (extra credit, up to 3 scant points)

Option 1: tag clouds. Use the results from your program to create a pretty or interesting tag cloud. For example, the cloud below was created using Wordle (TM) by supplying weights from the text of Jane Austen's *Sense and Sensibility* novel. To get started I suggest removing the 50 most common words (i.e., stopwords), using collection frequency as a weight, and providing around 200 words. I used the tool at: http://www.wordle.net/advanced (you may use this tool, or another). Wordle's format looks like this:

her:2551
she:1613
elinor:685
mrs:530
which:593

Your results will vary, depending on if you remove stopwords, how you define what is a word, how many words you use, which text corpus you use, any options used by the tool you use, etc... You can try other texts besides Yelp/Headlines.



Option 2: Plot the Zipfian distribution. Make a plot of document frequency on the vertical axis and word rank (by document frequency) on the horizontal. Ideally a log-log plot. You should see something similar to Figure 5.2 in the course text.

Note: you can both make a tag cloud and plot the Zipfian curve, but 3 bonus points is the maximum you're going to see.