SIMS 202 Information Organization and Retrieval: Assignment 9 – Text Processing and Zipf Distribution

assignment authors: Marti Hearst & Ray Larson

Overview

Assigned 10/31, Due 11/7.
The goal of this assignment is threefold. 1) It will introduce you to Unix systems, 2) it will give you a better idea of how content analysis of text documents is performed and 3) it will permit you to examine the Zipf distribution of terms in text databases.

Reading: Modern IR, Ch. 2.1 - 2.5, 8.1 - 8.2, Cheshire paper (in reader)
Time estimates:

Preliminaries: if you know Unix basics, 10 minutes. Otherwise, 1 or 2 hours to learn Unix basics.
Tokenizing: 30 minutes (just reading and following instructions)
Examining the data: 30 minutes
Viewing and Analyzing the Zipfian distribution: 60-90 minutes.
Total: 2.5-3 hours for those who know Unix, 5-6 hours for those who do not.

Note that the last part of this assignment has to be done using a lab machine.

Introduction

In this assignment you will run a tokenizer program over a set of technical abstracts on the topic of Artificial Intelligence. The tokenizer makes use of a stop list and a stemmer in order to create an inverted list. You will also see the creating of Zipf distributions first hand.

Preliminaries

This assignment requires limited use of Unix commands. There is a tutorial available in the lab. Another can be found on the class web site.

Bring up a Unix shell using the SecureCRT program. You can work on info or irony (irony is a faster machine).

Create a new directory in your home directory. Let's say you call it “my-a9”. Make this your current directory (by running the Unix command: cd my-a6). Data files are in http://www.sims.berkeley.edu/courses/is202/f02/as9_data

Copy all the files in that directory into your directory. You can use Netscape's save file command but it can mess up the format, so it is better to just do the following from the Unix prompt in your directory my-a6:

cp /www/docs/courses/is202/f02/as9_data/* .

The asterisk is a wildcard matching anything, and period at the end means “the current directory” so the entire command means "copy all the files to the current directory."

Take a look at the data files, ait*.t

Question 1: How big is ait1.t? How many different articles are in this file? (hint: use grep followed by wc).

Tokenizing and Inverting the Text Files

We have made available a program called postdoc that does the following things:

1. tokenizes the text
2. stems the text, and
3. creates a postings file.

(Note: the program is set up to ignore terms that occur only one time in the collection.)

You are going to run postdoc on two different sizes of text data. The first has about 10 AIT documents and the second has about 5000.

To use postdoc, first edit the foarc file to tell postdoc which input and output files to use. The symbolic links made above let us pretend that all the files are in your local directory even though they aren't. However, the output files you produce will end up in your local directory. We are doing it this way because we want every person to produce their own output files.

For the first time through, set **FilesFile** to be *short-files.d* and **KwinvFile** to *short-kw-inv.d* and **DocsFile** to *short-docs.d* . You have to keep the tab in between the two fields. The three edited lines should look like:

```
FilesFile   short-files.d
DocsFile    short-docs.d
KwinvFile   short-kw-inv.d
```

Leave the other lines as they are.

Now run the conversion program. You have to run it from the directory with the files you've created. It uses the foarc file to tell it where to look for input files and where to place the output files. You run it by giving the full pathname of the program. Just run it from the Unix prompt, like this:

./postdoc

If you get a *segmentation fault* message it probably means you did not edit foarc correctly. This is just like running a regular Unix program like "cd" except for the standard programs the system knows where they are located (because your **.cshrc** file contains information about where programs are usually located). *If you don't understand this paragraph, don't worry about it.*

After the program finishes you should have an output file *short-kw-inv.d* in your directory my-a6.

Now you have to modify foarc again. The second time through make the following changes to foarc (remember you have to make sure there is a tab between the two fields):

```
FilesFile   long-files.d
DocsFile    long-docs.d
KwinvFile   long-kw-inv.d
```

Run postdoc again. It should take a long time to finish. When it is done you should end up with a new file called *long-kw-inv.d*

Now run the program without using a stop list, on the small data set. To do this, first create an empty file called empty.wrd. Then, in foarc, change

```
FilesFile   short-files.d
DocsFile    short-docs.d
KwinvFile   short-kw-nostop-inv.d
StopFile    empty.wrd
```

Run the program again. You should end up with a new file called short-kw-nostop-inv.d

Question 2: How is *short-kw-nostop.inv.d* different than *short-kw-inv.d*, both quantitatively and qualitatively?

Examining the Data

Look at *short-kw-inv.d*. This is an inverted index. The first column shows the stemmed word, the second column shows how many documents the stemmed word occurs in (docfreq), the third shows its raw frequency in the collection (termfreq). The remaining fields show the IDs of the documents that contain the term. This is structured as follows: How often the term occurs (call this tf). The number of documents in which the term occurs tf times. The ids of those documents. In other words, the line:

```
commit  2     5     4 1 10  1 1 7
detail  3     3     1 3 2 4 7
```

means that the term commit occurred in two documents in the entire collection. Its total frequency (number of occurrences) in the collection was 5. In one document, it occurred four times. The document in which this happened was

document 10. In one document, it occurred one time. This document was document 7. The word detail occurred in 3 documents, for a total of 3 occurrences and it occurred 1 time in 3 documents, which are documents 2, 4, and 7.

Question 3: Describe, in the same, manner as above, the information associated with the term intellig for *short-kw-inv.d*.

Question 4: What are the five most frequent no-stopword terms in short-files? (Hint: use the Unix sort command on *short-kw-inv.d*.)

Viewing and Analyzing the Zipfian distribution

We now want to convert these inverted files into a form that can be viewed according to its Zipfian distribution. We are going to view the results using an information visualization program. Recall that the Zipfian distribution is an effect seen when the data is ordered by its rank. First we have to see how often each term occurs. The term that occurs most frequently is assigned rank 1. The term that occurs second most frequently is assigned rank 2, etc. Thus if frog occurs 100 times, and this is the most frequent word, it is assigned rank 1. If toad occurs 96 times, and this is second most frequent, it is assigned rank 2. Say three terms remain, and they all occur 2 times each. They are tied, but we deal with ties arbitrarily, assigning them increasing ranks, 3, 4, and 5.

Rather than writing a program, we can convert this file using a pipeline of Unix commands. It took me a little while to figure out how to make this work, so I am giving the sequence of commands to you here for converting *short-kw-inv.d:*

```
cat short-kw-inv.d | cut -f 1,3 | sort -k 2,2 -nr | \

awk '{s += 1; print s, "\t", $2, "\t", $1}' > short-zipf.txt
```

We are going to use *short-zipf.txt* as input to the visualization program, and it requires tab-separated input.

Question 5: Describe what each stage of the pipeline does. Hint: you can produce intermediate temporary results to see what each step does, e.g.,

```
cat short-kw-inv.d | cut -f 3 > out
```

and then look at the contents of "out".

Run the same command *on long-kw-inv.d* but put the results on *long-zipf.txt*. Do the same for *short-kw-nostop-inv.d* putting the results in *short-nostop-zipf.txt*.

Now run the visualization program. You have to use the SIMS machines for this as this is the only place it resides. The program is located at:

Start\Programs\Research & Analysis\Data Visualization\Spotfire Pro

Load in your data file short-zipf.txt using the File\Open menu choice (you have to tell it to take a file ending in .txt). The system should just load things properly. It shows the data as a scatter plot of rank against frequency.

Question 6: Using the visualization to answer this question: How many terms occur 10 times in *short-zipf.txt*? Which terms occur 15 times?

Question 7: Use the tabs on the X-axis and Y-axis to change the input the scatter plot. Set Y to Column 1 and X to Column 2. Does this change the shape of the curve very much? Now Change Y back to Column 2 and set X to Column 3. Does alphabetical order of terms correlate to word frequency?

Question 8: Now load in the data for *short-nostop-zipf.txt*. How is this different or similar to the graph for *short-zipf.txt*. Why?

Now load in the data for *long-zipf.txt*.

Question 9: Play around with the sliders to see different subsets of the data. Note that these are special sliders that let you focus on a subset of the data. Moving both arrows close together on an axis greatly reduces the number of data points you see. Adjust the sliders so you can answer this: about how many terms have frequencies between 480 and 500?

Question 10: (This is the important question.) Compare the fully-expanded initial view of long-zipf with the fully-expanded initial view of short-zipf. Discuss the similarities/differences. Why does this occur?

Question 11: (This is the other important question.) Think about tf*idf term weighting. Discuss how these graphs show why we divide the frequency with which a term occurs in an individual document (tf) by the number of documents it occurs in (df). Why is this a good strategy for ranking documents?

Question 12: Turn in screenshots of the visualization showing short-zipf and long-zipf (separate images are fine).

Please submit a hard copy of your assignment in class. Please clearly separate your answers from the questions and from each other to facilitate grading. Also, keep an electronic copy of your assignment in case we ask you to submit it to be posted as an example of a good answer!