Assignment 2

Fall 2017
CS834 Introduction to Information Retrieval
Dr. Michael Nelson

Orkun Krand

October 14, 2017

1 Question 4.1

1.1 Question

Plot rank-frequency curves (using a log-log graph) for words and bigrams in the Wikipedia collection available through the book website (http://www.search-engines-book.com). Plot a curve for the combination of the two. What are the best values for the parameter c for each curve?

1.2 Methodology

I wrote a Python script `pagevisitor.py` that uses NLTK (Natural Language Toolkit) and BeautifulSoup to access a folder and its subfolders (looking for 'en' as default as it is the root folder of the Wikipedia corpus from the book website), and process the documents found to collect word, bigram, and inlink information. By the time I realized I should've converted everything to lowercase to avoid duplicates, it was too late. `q1graphs.r` was created to visualize the data collected by `pagevisitor.py`

1.3 Results

`pagevisitor.py` found 232,919 words and 1,662,253 bigrams. Visualizing these numbers, we receive the following graphs:

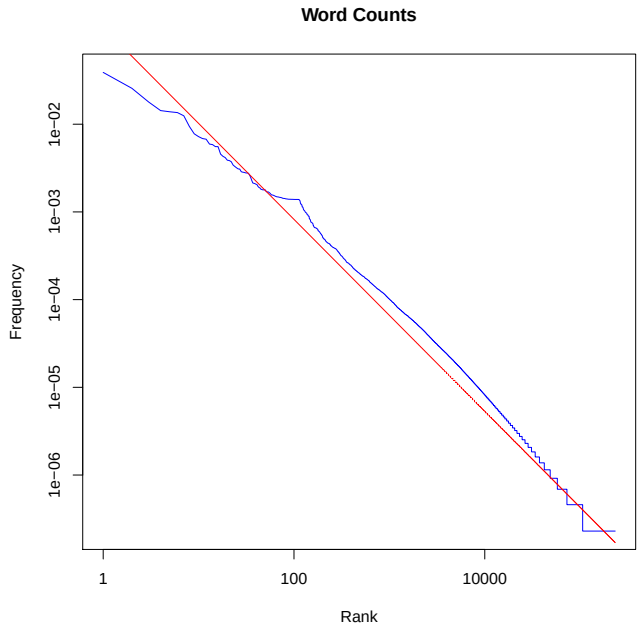

Figure 1: Word Count

Maximum likelihood estimation of words

Call: mle(minuslogl = ll, start = list(s = 1))

Coefficients: Estimate Std. Error s 1.002823 0.0001287333

-2 log L: 73354417

C = 1.269302e-01

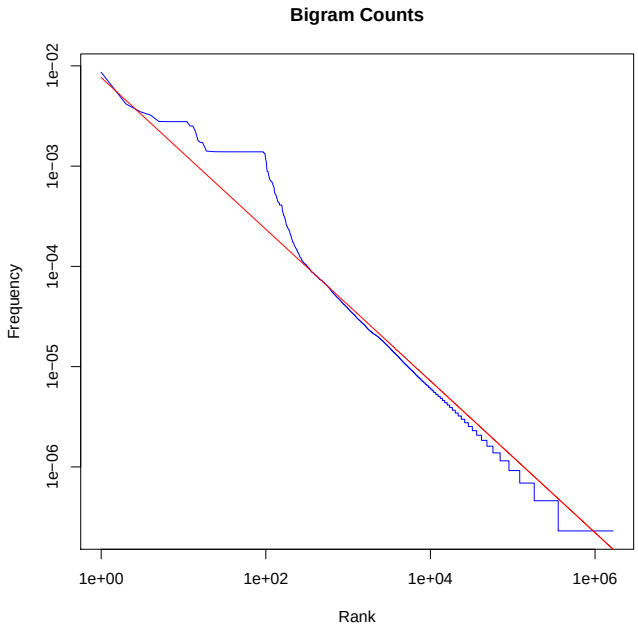

Figure 2: Bigram Count

Maximum likelihood estimation of bigrams

Call: mle(minuslogl = ll, start = list(s = 1))

Coefficients: Estimate Std. Error s 0.8292686 0.0001292527

-2 log L: 106198894

C = 7.650163e-03

Maximum likelihood estimation of combined

Call: mle(minuslogl = ll, start = list(s = 1))

Coefficients: Estimate Std. Error s 0.9250651 8.066917e-05

-2 log L: 191232526

C = 3.019035e-02

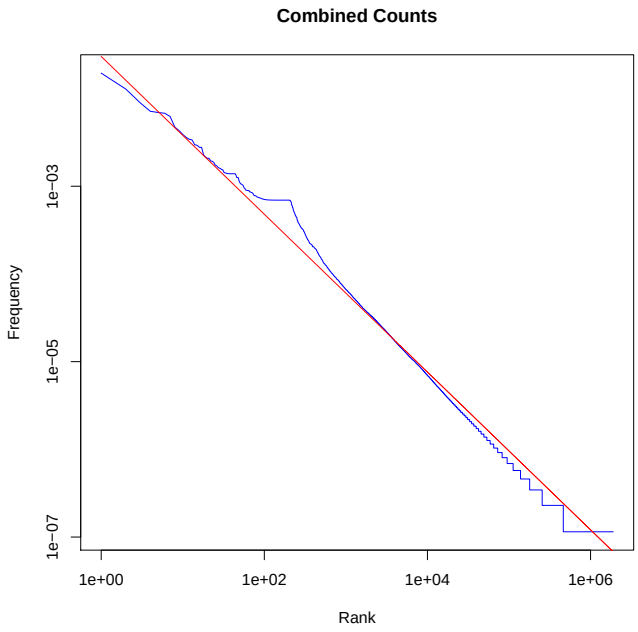

Figure 3: Both Word and Bigram Counts

2 Question 4.2

2.1 Question

Plot vocabulary growth for the Wikipedia collection and estimate the parameters for Heaps' law. Should the order in which the documents are processed make any difference?

2.2 Methodology

Modifying `pagevisitor.py` to keep track of vocabulary (unique words) and total words and logging them after accessing each document enabled me to create the required plot for this question. Using `q2graphs.r`, I was able to create visualizations.

2.3 Results

The order in which the documents are processed is important for Heap's Law because we compute the sum of all words in the dictionary after processing each document.

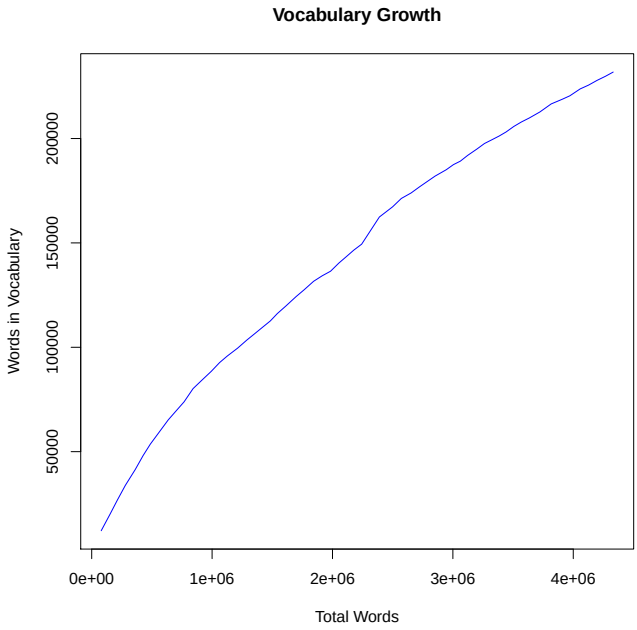

Figure 4: Vocabulary Growth

3 Question 4.3

3.1 Question

Try to estimate the number of web pages indexed by two different search engines using the technique described in this chapter. Compare the size estimates from a range of queries and discuss the consistency (or lack of it) of these estimates.

3.2 Methodology

According to the book[1], assuming the two terms are independent of each other,

$$f_{ab} = N \cdot f_a/N \cdot f_b/N \tag{1}$$

$$N = (f_a \cdot f_b)/f_{ab} \tag{2}$$

I will use Google and Yahoo! for this assignment. The queries I will use are "motorcycle cake" and "yamaha quinoa".

3.3 Results

Working with Google first:

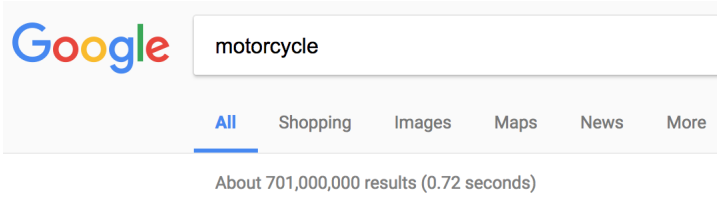

Figure 5: Google: motorcycle

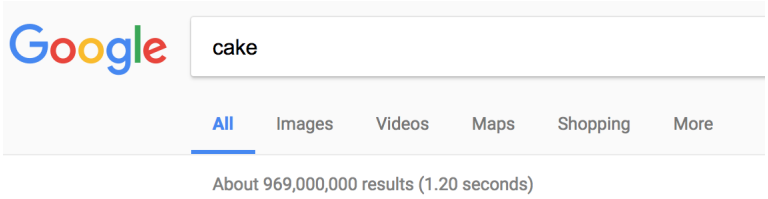

Figure 6: Google: cake

According to the first query, the size of Google is:

$$N = \frac{(701,000,000 \cdot 969,000,000)}{18,100,000} = 37,528,674,033 \tag{3}$$

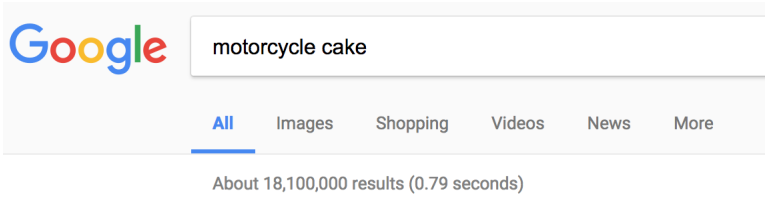

Figure 7: Google: motorcycle cake

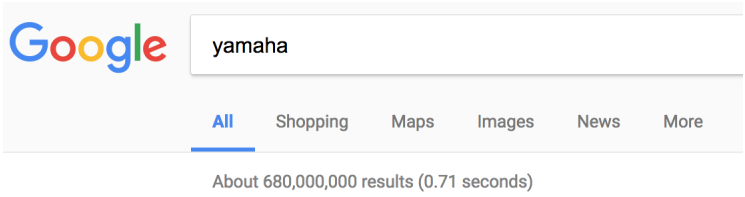

Figure 8: Google: yamaha

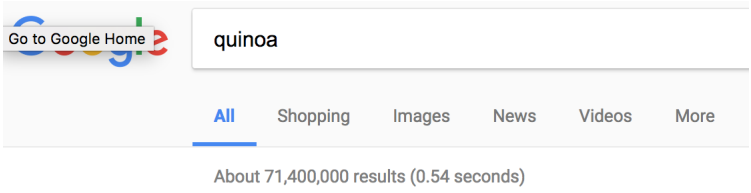

Figure 9: Google: quinoa

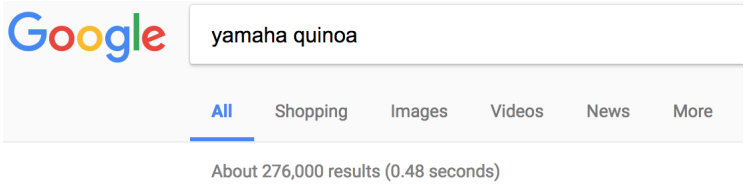

Figure 10: Google: yamaha quinoa

The second query, using the equation in the book gives us:

$$N = \frac{(680,000,000 \cdot 71,400,000)}{276,000} = 175,913,043,478 \tag{4}$$

These two numbers are very far away from each other. This may be due to *motorcycle* and *cake* both being generic terms whereas *yamaha* and *quinoa* are more specific terms that are not related. So if I had to pick between these two results, I would pick the larger (176B) as the better estimation for the size of Google.

Let's do the same for Yahoo!:

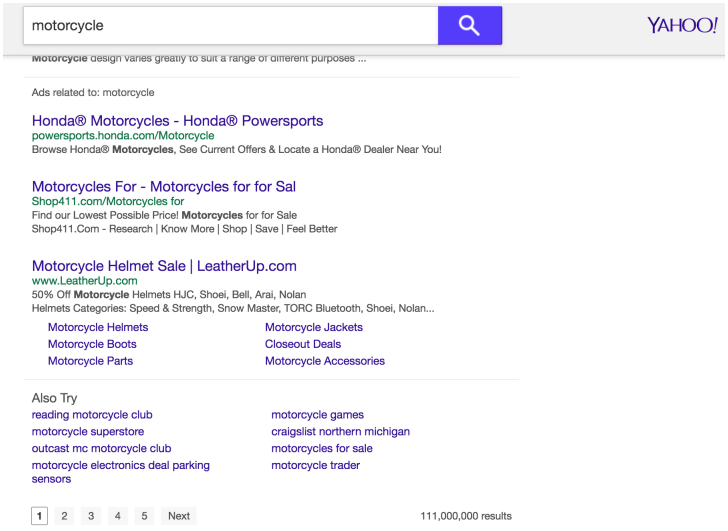

Figure 11: Yahoo: motorcycle

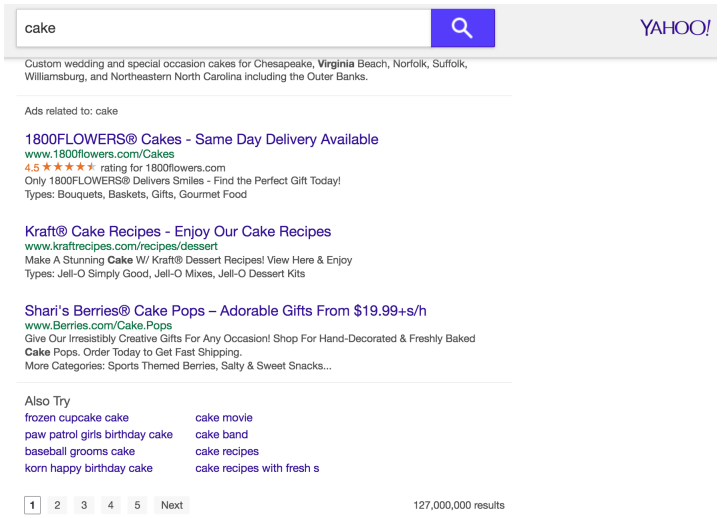

Figure 12: Yahoo: cake

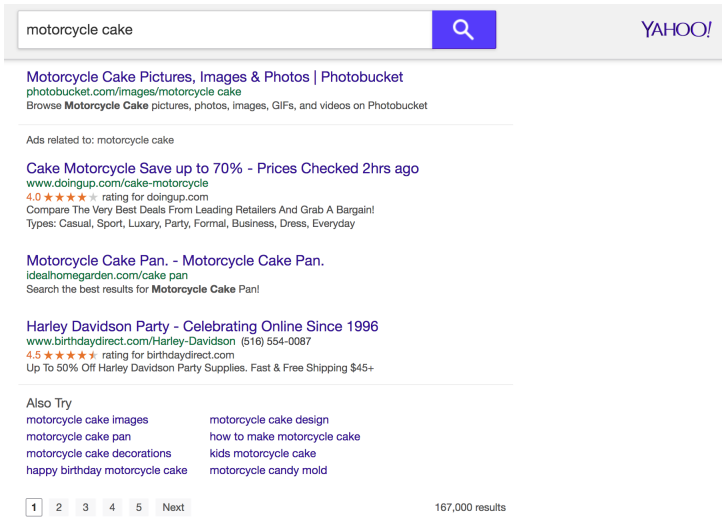

Figure 13: Yahoo: motorcycle cake

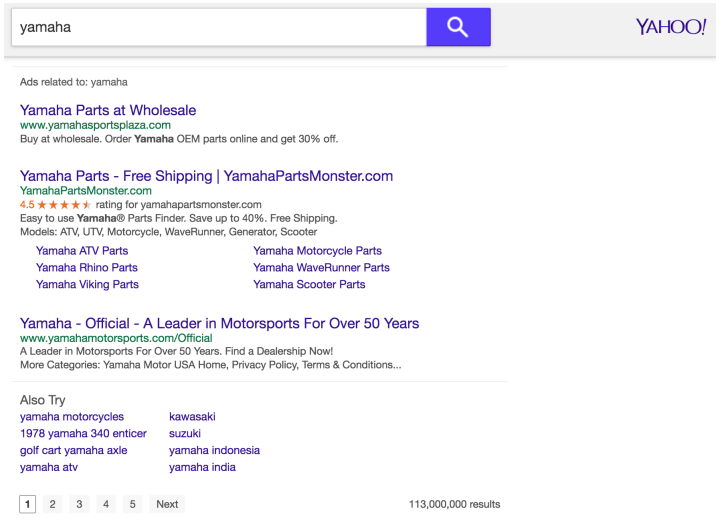

Figure 14: Yahoo: yamaha

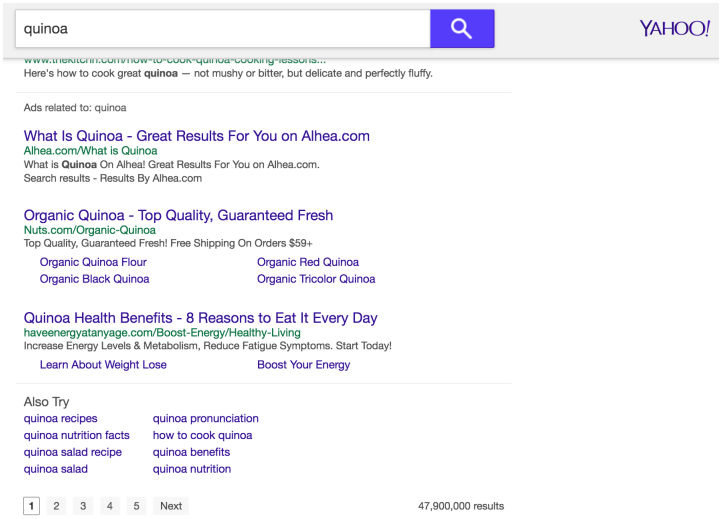

Figure 15: Yahoo: quinoa

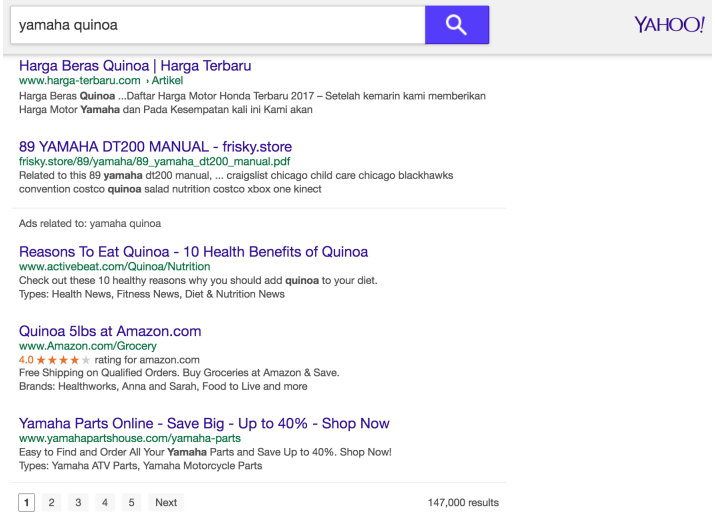

Figure 16: Yahoo: yamaha quinoa

According to the first query, the size of Yahoo! is:

$$N = \frac{(111,000,000 \cdot 127,000,000)}{167,000} = 84,413,173,652 \tag{5}$$

The second query, using the equation in the book gives us:

$$N = \frac{(113,000,000 \cdot 47,900,000)}{147,000} = 36,821,088,435 \tag{6}$$

Unlike the first case, the second equation gave a smaller number than the first. But I still find the second number more believable because of the reason I explained earlier. This would mean that Google is almost 5 times bigger than Yahoo!. Checking out the real numbers surely would be interesting.

4 Question 4.8

4.1 Question

Find the 10 Wikipedia documents with the most inlinks. Show the collection of anchor text for those pages.

4.2 Methodology

Modifying `pagevisitor.py` to find the inlinks for every page accessed was required for to find the top 10. Anchor text was also added to the output. Sorting them based on the number of inlinks gave me the top 10 pages.

4.3 Results

Number of Inlinks	URI	Anchor Text
2264	articles/2/0/0/2007.html	2007, As of 2007
1896	articles/s/m/a/User%7ESmackBot_cc7a.html	SmackBot
1770	articles/2/0/0/2008.html	2008
1363	articles/u/n/i/United_States_09d4.html	United States Of America, USA, Union, Thirteen Colonies, U.S., United States of America, US, United States, American, Americans, America, American nation, American citizen
982	articles/2/0/0/2006.html	2006
791	articles/a/l/a/User%7EAlaibot_de3d.html	Alaibot
676	articles/c/y/d/User%7ECydebot_38a6.html	Cydebot
675	articles/l/i/v/Category%7ELiving_people_7259.html	Living people
663	articles/b/l/u/User%7EBluebot_e595.html	Bluebot
655	articles/g/e/o/Geographic_coordinate_system.html	coordinates, Location, Coordinates

Table 1: Top Ten Pages With the Highest Number of Inlinks

5 Question 5.8

5.1 Question

Write a program that can build a simple inverted index of a set of text documents. Each inverted list will contain the file names of the documents that contain that word. Suppose the file A contains the text "the quick brown fox", and file B contains "the slow blue fox".

```
The output of your program would be:
% ./your-program A B
blue B
brown A
fox A B
quick A
slow B
the A B
```

5.2 Methodology

Modifying the `pagevisitor.py` script allowed me to create an inverted index of the Wikipedia corpus. I wasn't sure what you meant by examples from the data set so I went ahead and created an inverted index for the whole thing. It ended up being a big file (104MB) but oh well.

References

[1] Croft, William Bruce, et al. *Search Engines: Information Retrieval in Practice*. Pearson, 2010.