Problem One

a.Write a quick program or script that counts word frequencies. For the most frequent 25 words and for the most frequent 25 additional words that start with the letter f (a total of 50 words), print the word, the number of times it occurs, its rank in the overall list of words, the probability of occurrence, and the product of the rank and the probability. Also indicate the total number of words and the total number of unique words that you found. Discuss whether this text satisfies Zipf's Law. Feel free to use other parts of the ranked list of terms.

Answer:
script used: p1.py
p1.txt shows the result for this problem.
According to file product.txt, we can find that r * Pr = constant, which is around 0.1.But for those words with very high and low frequency,the value varies a little more, which is fine as Zipf's law works not that well in this condition.So in conclusion the result I get satisfies Zipf's law.

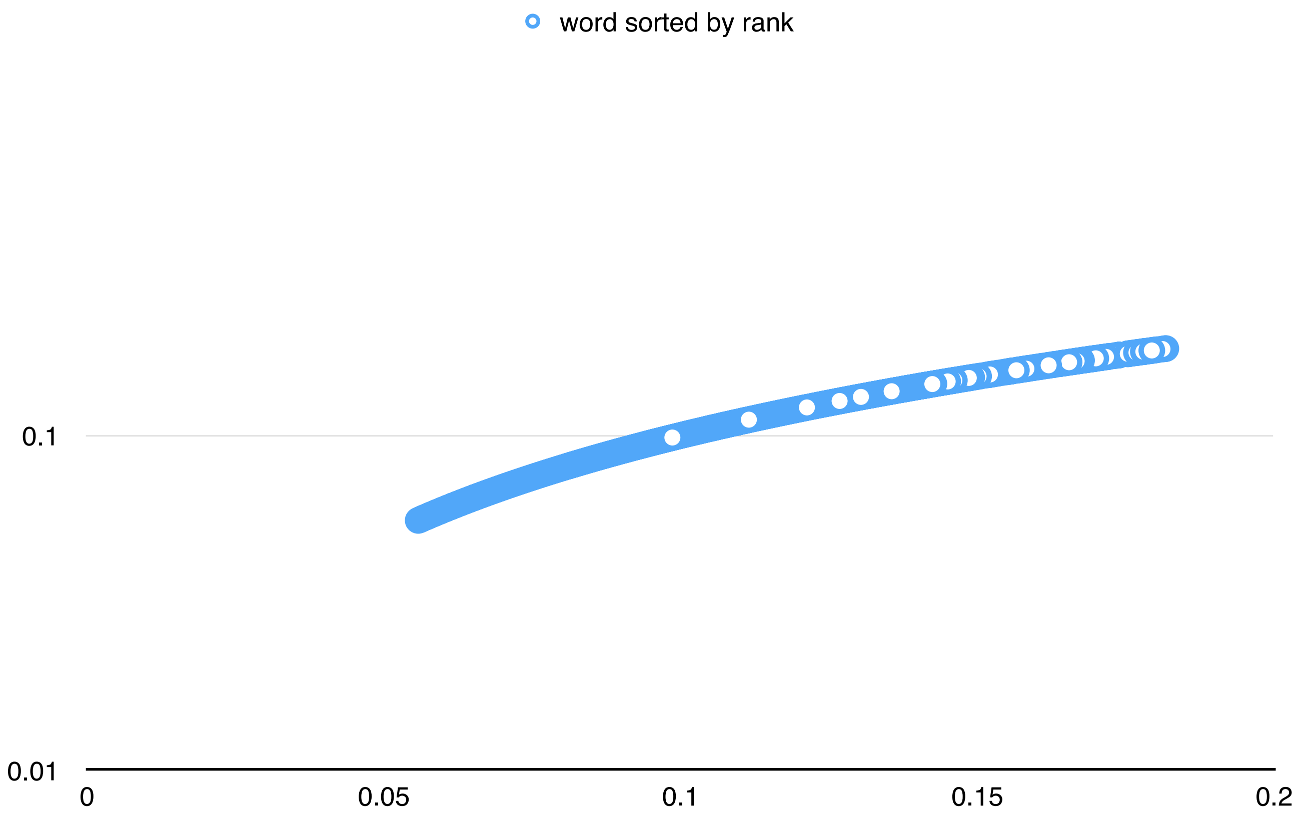

Chart shows the log(rank*prob) trend, which also prove the conclusion

b.Suppose that while you were building retrieval index, you decided to omit all words that occur fewer than five times (i.e., one to four times). According to Zipf's Law, what proportion of the total words in the collection would you omit? (Be ready to justify your answer.) What proportion would actually be omitted in the Alice in Wonderland text above?

Answer:

According to Zipf's Law, word that occurs n times has rank $r_n = k/n$,and the number of words with frequency n is $r_n - r_{n+1} = k/n - k/(n + 1) = k/n(n + 1)$

Proportion found by dividing by total number of words = highest rank = k So, proportion with frequency n is 1/n(n+1).

By this we can compute that proportion would be omit is 1/2 + 1/6 + 1/12 + 1/20 = 0.8

The actually omit proportion in Alice's Wonderland is about .74. But I have some extra thoughts of this figure, what if some less frequency words are usually combined with other words to make sense(like "york","new" and "New York").And what if words like "crusade" and "Crusade" are very different with first word in capital? To me, the real number would be less than .74 if we do more dig up on the document.

Problem Two

a.According to Heaps' Law, what proportion of a collection of text must be read before 90% of its vocabulary has been encountered? You may assume that beta=0.5. Hint: to solve this problem you don't need to know the value of K.

Answer:

Assume beta = 0.5, so v = k * n ^ 0.5. Then we get 0.9 * v = 0.9 * k * n ^ 0.5 = (0.81 * n) ^ 0.5 * k

Conclusion: 81% of collection of text must be read before 90% of its vocabulary has been encountered

b.[Extra credit] Verify Heap's Law on the Alice in Wonderland text. Process each word of the text, in order, and compute the following pairs of numbers: (number of words processed, number of unique words seen). These pairs of numbers, treated as (x,y) values, should satisfy Heaps Law. Appropriately transform the data and use least squares to determine the model parameters K and beta. As explained in class, there are better ways of estimating a power-law distribution, but this quick-and-dirty hack is often effective.

Answer:

See my script p2.py, which gives beta = 0.61406957582, k = 5.28012077419

according to Heap's law v = k * n ^ β => log v = log k + β*log n,assume x = log n,y = log v, k2= log k'

we got y = k' + β x, and do linear regression with least squares to determine k2 and β, k = pow(10, k')

Problem Three

Use the result numbers associated with a web search engine query to do the following:

Think up two 3-word queries and submit them to a search engine. Tell us which search engine you used. Estimate the number of results for these two queries using numbers from single words and pairs of words contained in the query. Compare them to the numbers returned by the search engine. Be ready to discuss results.

Estimate the size of the search engine's indexed corpus. Compare the size estimates from the two queries and discuss the consistency (or lack of it) of these estimates.

Answer:

Part 1:

I use Google here.

Word 1: Messi best goals

which google tells me he find 34,500,000 results.

Trying with single word :

best - 2,390,000,000 messi - 87,200,000 goals - 135,000,000

Trying with 2-word combinations

Messi best - 105,000,000 best goals - 90,300,000 Messi goals - 38,400,000

f(Messi∩best∩goals) = f(Messi∩best) * f(messi∩goals) / f(Messi)
= 4,032,000,000,000,000 / 87,200,000

= 46,000,000

Seems like result is a bit far from the real value. Let me try another one

Word2 : famous Golf players

which google gave me 26,700,000 results

Trying with single word:

famous - 224,000,000 Golf - 428,000,000 players - 252,000,000

Trying with 2-word combinations

famous golf - 201,000,000 famous players - 36,400,000 golf players - 244,000,000

f(famous∩Golf∩players) = f(famous∩Golf) * f(Golf∩players) / f(Golf)

= 201,000,000 * 244,000,000 / 428,000,000

= 115,000,000

Further discussion and doubts :

Did same analysis to 2 phrases and both results far from the real value. The formula seems doesn't work well with Google. I think when we do single word search, mostly it's just a wildcard try, it's easy to give you the numbers(just the length of the inverted list).However, when query words get complicated,the search engine is more likely to rank a few thousand documents and provide a number based on what it found there.

"This method assumes that documents containing target query string distributes evenly through all documents."It's what I got after some dig up. But it isn't always this perfect, which explains why the theory result so different from real value. Google should be using a more advanced algorithm to do this.

Part 2:

N = (f Messi * f best goals)/f Messi best goals

= (87,200,000 * 90,300,000) / 34,500,000

= 2,790,000,000

N = (f famous * f Golf players)/f famous Golf players

= 224,000,000 * 244,000,000 / 26,700,000

= 2,050,000,000

We have 2 different values now, so to me it's not a consistent estimate.

1) In the formula, f refers to the frequency of word in corpus, instead we use the number of result returned from search engine

2) According to the formula statement, word a and b should be independent, which Messi satisfied but Golf not that good, that maybe another reason why the result varied.

Problem Four

Script p4.py is my code to this.

a. answer is in bigram_freq.txt, just simply modify q1 script

b. top_ten_bigram.txt is the desired answer, got same issue mentioned in question c,and solved together after asking Mat.

c. p4_3.txt shows my calculation for this question.

Comments: should consider words like : 'the said' and 'said the' as same word occur twice when deal with 'said' in phrase ' the said the' .