Sheetal Parikh
Problem Set – Module 3
605.744.81 Information Retrieval
Fall 2021

1.) Qualitatively explain how the use of stemming impacts the following:
(a) vocabulary size;
(b) the total number of postings in an inverted file
(c) the average (over all terms) of posting list length?

Stemming decreases vocabulary size by approximately 17% since it cuts off the ends of words down to their root. Therefore, many words that would have been considered different terms, would be now be considered the same word.

In an inverted index, each term has a postings list or a list of documents that contain the word and the frequency of that word. A posting is an individual entry in the list, meaning an entry for a docid and the frequency of a term. Because stemming reduces the vocabulary size, the number of postings would decrease as well because the number of words that would have postings lists would decrease. After stemming, words that were once considered different in a document may be considered the same word and would be reflected in that document's frequency counts. The reduction in the number of postings, however, would be less than the decrease in vocabulary size. For example, the reduction in the number of non-positional postings would be approximately 4%. Stemming may cause certain terms to have additional postings added to their postings list because additional documents now recognized having that word after stemming. The lower rate of reduction in postings could be due to these new postings being added to a list.

The average posting list length would roughly stay the same because overall the terms would approximately be found in the same number of documents. The stemmed terms would be found in a similar number of documents as before but have different frequency counts. Also, if stemming caused any postings to be removed from or added to a list, the length of the postings lists even after these changes would average out to be the same as before stemming.

2.) Express these three numbers {41, 128, and 500} in three ways: (a) using a fixed 12-bit binary representation, (b) with gamma coding, and (c) with delta coding. You must follow the method for computing gamma/delta described in the text and presented in the lecture materials. I recommend learning to do this by hand, but you may write (and provide) a short computer program if you prefer – but do not use a program that you did not write yourself.

a.) *12-bit binary representation*

I. $(41)_{10} = (0000\ 0010\ 1001)_2$
   $41 = 2^0 + 2^3 + 2^5$
II. $(128)_{10} = (0000\ 1000\ 0000)_2$
   $128 = 2^7$
III. $(500)_{10} = (0001\ 1111\ 0100)_2$
   $500 = 2^2 + 2^4 + 2^5 + 2^6 + 2^7 + 2^8$

b.) *Gamma coding*

I. 41 = unary(5) followed by binary(9) = 111110 01001
   $\log_2(41) = 5$ → unary (5) = 111110
   $41 - 2^5 = 9$ → binary (9) = 01001

II. 128 = unary(7) followed by binary(0) = 11111110 0000000

$\log_2(128) = 7 \rightarrow$ unary (7) = 11111110
$128 - 2^7 = 0 \rightarrow$ binary (0) = 0000000

III. 500 = unary(8) followed by binary(244) = 111111110 11110100
$\log_2(500) = 8 \rightarrow$ unary (8) = 111111110
$500 - 2^8 = 0 \rightarrow$ binary (244) = 11110100

c.) Delta Coding

I. 41 = gamma(5) followed by binary(9) = 110 01 01001
$\log_2(41) = 5 \rightarrow$ gamma(5) = 110 01
$41 - 2^5 = 9 \rightarrow$ binary(9) = 01001

II. 128 = gamma(7) followed by binary(0) = 110 11 0000
$\log_2(128) = 7 \rightarrow$ gamma(7) = 110 11
$128 - 2^7 = 0 \rightarrow$ binary(0) = 0000

need 7 suffix bits so 0 should be 0000000

III. 500 = gamma(8) followed by binary(244) = 1110 000 11110100
$\log_2(500) = 8 \rightarrow$ gamma(8) = 1110 000
$500 - 2^8 = 0 \rightarrow$ binary(244) = 11110100

3.) Below is a bit sequence for a gamma encoded gap list (as described in Chapter 5 of IIR and the lecture materials). Decode the gap list and reconstruct the corresponding list of docids. Spaces are added for ease of reading and note that the final part only has two bits. Hint: there are four docids.
1111 1100 0100 0111 1111 0010 0000 1111 1010 1011 1111 0100 00

1111 1100 0100 0111 1111 0010 0000 1111 1010 1011 1111 0100 00

We can section off the bit sequence into the docids. Since we have a gamma encoded gap list we can count the number of one bits until we find a 0. The one bits and the extra 0 bit are the unary code. The following bits after the 0 would be the same length as the number of one bits. This would be the binary code.

Docid	72	232	285	333
Gap		160	53	48
Gamma encoded	1111 1100 0100 0	111 1111 0010 0000	1111 1010 101	1 1111 0100 00
	unary(6) followed by $(001000)_2$	unary(7) followed by $(0100000)_2$	unary(5) followed by $(10101)_2$	unary(5) followed by $(10000)_2$

1st gap = 111111 0 001000 = unary(6) followed by 001000 = $2^6 + 2^3 = 72$
docid #1 = 72
2nd gap = 1111111 0 0100000 = unary(7) followed by 0100000 = $2^7 + 2^5 = 160$
docid #2 = 72 + 160 = 232
3rd gap = 11111 0 10101 = unary(5) followed by 10101 = $2^5 + 2^4 + 2^2 + 2^0 = 53$
docid #3 = 232 + 53 = 285
4th gap = 11111 0 10000= unary(5) followed by 10000 = $2^5 + 2^4 = 48$
docid #4 = 285 + 48 = 333

4.) Given a document collection of 1 billion words (or tokens), use Heap's Law to calculate the expected vocabulary size to the nearest 1,000 terms. How would your answer change if the collection were 100 times larger? For this problem use parameters k=50 and b = 0.5.

a.) $M = kT^b$
T = tokens = 1 billion
K = 50
B = 0.5

$M = kT^b = 50(1{,}000{,}000{,}000)^{0.5}$
= 1,581,138.83
≈ 1,581,139
≈ 1,581,000 Terms

Expected vocabulary size to the nearest 1,000 terms when we have 1 Billion Tokens = 1,581,000 Terms

b.) $M = kT^b$
T = tokens = 100 billion
K = 50
B = 0.5

$M = kT^b = 50(100{,}000{,}000{,}000)^{0.5}$
= 15,811,388.3
≈ 15,811,000 Terms

Expected vocabulary size to the nearest 1,000 terms when we have 100 Billion Tokens = 15,811,000 Terms
We see that if the collection is 100 times larger than the vocabulary size will be 10 times larger

5.) Below is a bit sequence for a set of gaps encoding using Variable Byte encoding as described in Chapter 5 of IIR. Decode the list of gaps and reconstruct the corresponding list of docids. Hint: there are four docids.
1000 0001 0000 0010 0001 1111 1000 0011 0001 0001 1000 0011 0000 1011 1010 1110

1000 0001 0000 0010 0001 1111 1000 0011 0001 0001 1000 0011 0000 1011 1010 1110

We can section off the bit sequence into the docids. Since we have a Variable Byte encoded gap list we know that list is organized into 8 bits at a time in which the first bit tells us whether we should stop with the current 8 bits (if the first bit is a 1) for the gap or if should continue to the next 8 bits (if the first bit is a 0).

Docid	1	36,740	38,919	40,373
Gap		36,739	2,179	1,454
VB Code	1000 0001	0000 0010 0001 1111 1000 0011	0001 0001 1000 0011	0000 1011 1010 1110
Binary Code	000 0001	000 0010 001 1111 000 0011	001 0001 000 0011	000 1011 010 1110

1st gap = $(000\ 0001)_2 = (1)_{10}$
docid #1 = 1
2nd gap = $(000\ 0010\ 001\ 1111\ 000\ 0011)_2 = (36{,}739)_{10}$
docid #2 = 1 + 36,739 = 36,740
3rd gap = $(001\ 0001\ 000\ 0011)_2 = (2{,}179)_{10}$

docid #3 = 36,740 + 2,179 = 38,919

4th gap = $(000\ 1011\ 010\ 1110)_2$ = $(1{,}454)_{10}$

docid #4 = 38,919 + 1,454 = 40,373