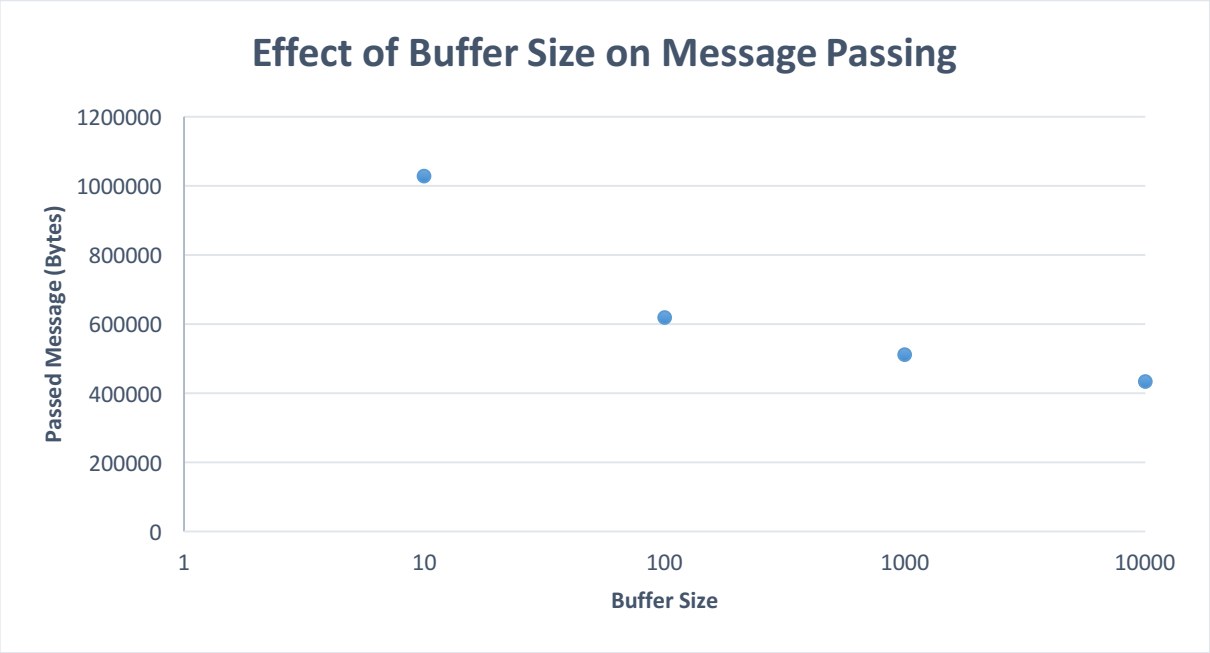
10-605 Assignment 1 Report: Streaming Naïve Bayes

Question 1:

Buffer Size	Passed Message File Size
10	1027872
100	617627
1000	510942
10000	433830



On average, the size of passed message decays at a rate of 0.6 with the logarithm of the buffer size.

Question 2:

1. Using the output stream of MergeCounts as input stream, we can first extract the words in each of the messages (i.e. W = w') and output the words to output stream.
2. Then, we can utilize the Unix command "sort" to sort the output stream that each line starts with W = w'.
3. Using the sorted messages as input stream, we can then count the size of vocabulary by increasing the count whenever we observe a difference in the word of the current line and the word of the previous line.

Question 3:

1. Initialize a counting number for |D|
   For each line of input

a. first separate document id from document content
b. for each word in document content, output a message:
   (word, doc_id) (tf=1, idf=1)
c. increase count of |D| by 1.

Do this for all lines of input.
Output the message "#D= count of |D|". (This is because '#' comes before '(' in ASCII table and would be sorted to the first place in the next step.)
Input of step 1:

d1 \t words…
d2 \t words…
…

Output of step 1:

(word1, d1) (tf=1, idf=1)
(word2, d1) (tf=1, idf=1)
(word1, d1) (tf=1, idf=1)
…
(word1, d2) (tf=1, idf=1)
…

2. Use Unix sort command, sort the output of step 1.
   Input of step 2: Output of step 1.
   Output of step 2:

#D= count of |D|
(word1, d1) (tf=1, idf=1)
(word1, d1) (tf=1, idf=1)
(word1, d2) (tf=1, idf=1)
…
(word2, d1) (tf=1, idf=1)
…

3. Scan first line and create a local variable d for count of |D|.
   Merge lines with same word and document id by accumulating their tf values and keep track of number of different documents a word appears in, call it df.
   Whenever word/doc_id changes, output a message:

(word, doc_id) (tf=n, idf=d/df)
where n is the accumulated tf values of (word, doc_id) pair.

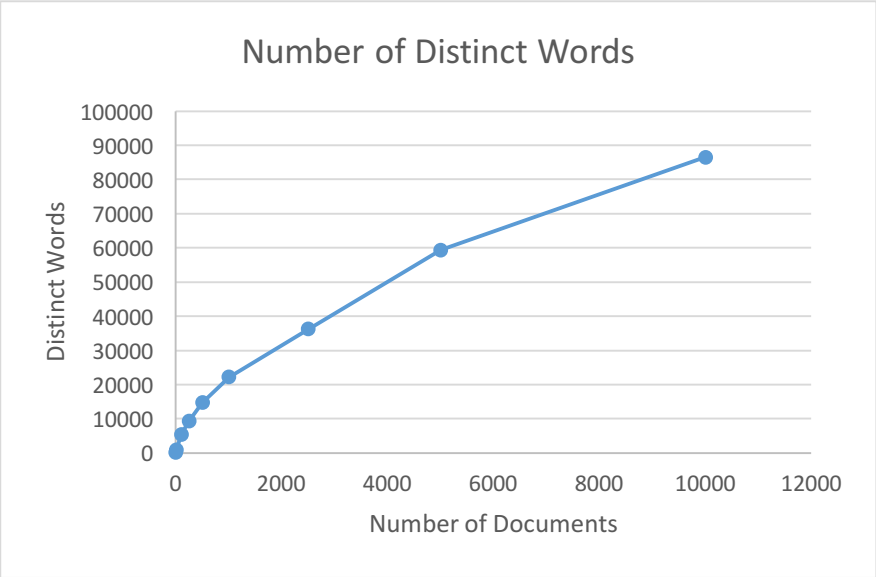
   Input of step 3: Output of step 2.
   Output of step 3:

(word1, d1) (tf=k1, idf=d)
(word1, d2) (tf=k2, idf=d/2)
(word2, d1) (tf=l1, idf=d)
…

4. Assuming that the total number of documents is not too large, scan through documents and save the doc_id and tf of the word in a list, and keep track of the last idf.
   Once the word changes during scanning, output messages:
   (word, doc_id1) (tf=k1, idf=last idf)
   (word, doc_id2) (tf=k2, idf =last idf)
   …
   Input of step 4: Output of step 3.
   Output of step 4:
   (word1, d1) (tf=k1, idf=idf1)
   (word1, d2) (tf=k2, idf=idf1)
   (word2, d1) (tf=l1, idf=idf2)
   …

Question 4:

Number of Documents	Number of Distinct Words
1	101
10	911
100	5314
250	9258
500	14720
1000	22173
2500	36195
5000	59318
10000	86451



From the graph, the number of distinct words is about proportional to the square-root of number of documents.