MSCI 541 Homework 4

Thomas Kleinknecht
Watiam: tkleinkn
Student ID: 20883814

Problem 1

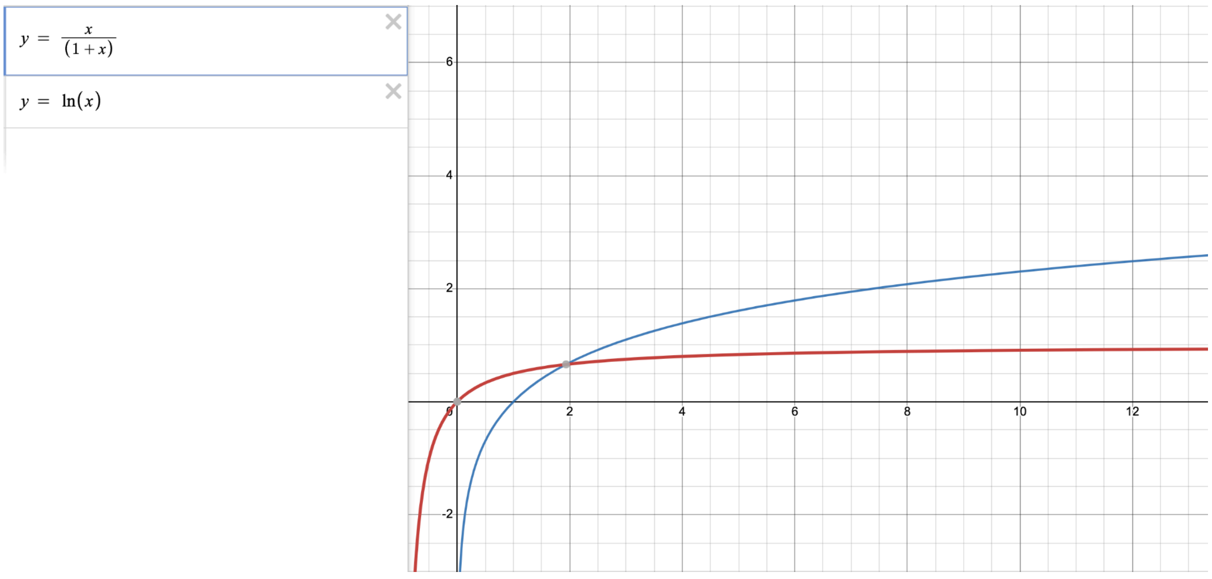
No score or metric can truly define the relevance of a document, as only the user knows their true information need and the context of the query. This means that, although a query can generate the same BM25 score, this is still only a measure of how often words appear in documents and is merely one way in which we estimate the relevance of the documents. It is, however, just an estimation.

Problem 2

a) The logarithm is used for term frequency to represent the decreasing amount of information which a higher frequency gives us with respect to the meaning of the document. It also helps to normalize for document length. This is because a document having 50 vs 60 appearances of a term tells us much less than a document having 1 vs 11 appearances of a term. Also, a document with 40 appearances of a term, twice as long as a document with 20 appearances of a term, is not twice as much "about" that term.

b) These same principles are achieved through BM25 in two ways:

   i) The structure of the equation: As BM25 has the term frequency as a ratio over itself + a constant, it has an asymptote at 1, and therefore has the same affect as the logarithm in reflecting the decreasing utility which the frequency has in determining the "aboutness" of a document. We can see this similarity illustrated below with a k value of 1 for this example:



   ii) The k value in the denominator, which in its formulation, uses that docs length vs. the average doc length in the collection, as well as the b parameter, to normalize for document length, handling the 40 vs. 20 appearance case from above.

Problem 3

If you wanted to use the query term frequency, you could multiply the BM25 score generated for that query term by the frequency of the term in the query. This would increase the contribution to the sum by that query term.

Problem 4

These query terms are typically handled by having no contribution to the score for each doc. As the score is comprised of the sum of the scores from each query term for each doc, the score will just be comprised of the sum of the scores for the other query terms.

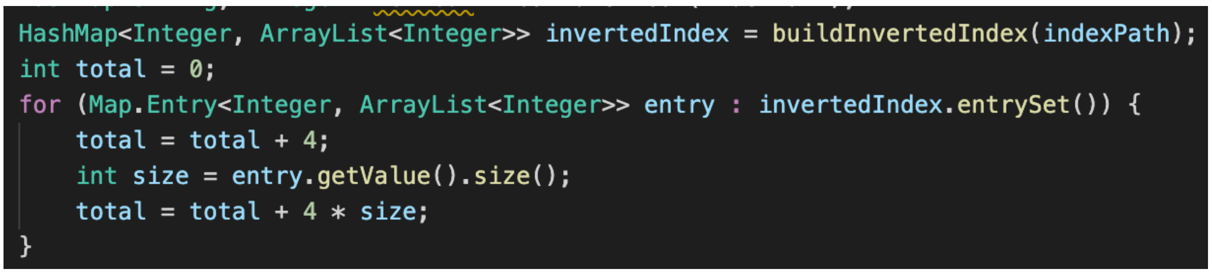
Problem 5

We can see from our original lexicon the number of unique words and from the doc id mapping file the number of docs after the tokenization from HW1.

numWords = 246970
numDocs = 131896

Therefore if each cell used 4 bytes, this would result in:
Memory (Matrix) = 246970 * 131896 * 4 = 130297420480 bytes = 130.297 gigabytes

However, for the inverted index, if each term ID takes 4 bytes, as well as each entry (doc ID or count), the below code returns a value of 256107448 bytes, or 0.256 gigabytes.



```
HashMap<Integer, ArrayList<Integer>> invertedIndex = buildInvertedIndex(indexPath);
int total = 0;
for (Map.Entry<Integer, ArrayList<Integer>> entry : invertedIndex.entrySet()) {
    total = total + 4;
    int size = entry.getValue().size();
    total = total + 4 * size;
}
```

This results in 130041313032 bytes, or 130.041 gigabytes of memory savings.

Problem 6

There are three core issues which are contributing to Gary's poor results:

1. No numbers are included in tokens or as tokens
2. He removes stopwords from the query terms but not the document terms in the collection
3. He stems query terms but not document terms in the collection

Of these three main issues, the one which has contributed the most to his poor retrieval performance is the stemming of query terms but not terms in the collection. This can cause many of the query terms to lose their original meaning, and even worse, Porter stemming can produce non-words, and if the same stemming technique was not used on the document collection, these words would never find a match. The next most impactful mistake is his removal of stopwords from the query but not the documents. This would result in less returned results for these queries, and therefore less results to rank and a lower recall as some potentially relevant documents will not be retrieved. The first point may have an impact, as some numbers, especially dates, can be critical to a subset of queries. These queries would have very few relevant results and a very low recall.

<u>Problem 7</u>
Comparing Performance:
The below values were generated from measures generated by the HW3 program on the baseline and stemmed results. The p-value was generated by the use of a two-sided, paired t-test with the two sets of data.

Run Name	Mean Average Precision	Mean P@10	Mean NDCG@10	Mean NDCG@1000
baseline	0.208242384	0.2533333	0.334475048	0.424264134
stem	0.249807706	0.2844444	0.37236682	0.484427443
p-value	0.006234725	0.1552383	0.115645437	0.002507866

As we can see above, given a standard p value threshold of 0.05, the difference in mean average precision and mean NDCG@1000 is statistically significant, however for the other two metrics, mean precision@10 and mean NDCG@10, the difference is not statistically significant.

In the below table, each topic is represented by its own row. The value is 1 if the stemmed measure is better than the baseline measure. On the two columns to the right, there is a count of how many metrics for each topic were improved, as well as whether or not it is determined to be better. If 3 or all 4 metrics were better, it says “YES” whereas if half, or 2, of the measures were better it says “MAYBE” or if 1 or none were better it says “NO”.

Comparison:	AP	Precision@10	NDCG@10	NDCG@1000	Metrics it is better	Better?
401	1	1	1	1	4	YES
402	1	1	1	1	4	YES
403	1	1	1	1	4	YES
404	1	0	0	1	2	MAYBE
405	0	0	0	0	0	NO
406	1	1	1	1	4	YES
407	0	0	0	0	0	NO
408	1	0	0	1	2	MAYBE
409	1	1	1	1	4	YES
410	1	1	1	1	4	YES
411	1	1	1	1	4	YES
412	1	1	1	1	4	YES
413	1	0	0	1	2	MAYBE
414	1	1	1	1	4	YES
415	1	1	1	1	4	YES
417	1	1	0	1	3	YES
418	1	0	0	1	2	MAYBE
419	1	1	1	1	4	YES
420	0	1	1	0	2	MAYBE
421	1	0	0	1	2	MAYBE
422	1	1	1	1	4	YES
424	1	0	0	1	2	MAYBE
425	1	1	1	1	4	YES
426	1	0	0	0	1	NO
427	1	1	1	1	4	YES
428	0	1	0	0	1	NO
429	1	1	1	1	4	YES
430	1	1	1	1	4	YES
431	1	0	0	1	2	MAYBE
432	0	0	0	0	0	NO
433	0	0	0	0	0	NO
434	0	1	1	0	2	MAYBE
435	0	0	0	0	0	NO
436	1	1	1	1	4	YES
438	1	0	0	1	2	MAYBE
439	1	0	0	1	2	MAYBE
440	0	0	0	0	0	NO
441	1	1	1	1	4	YES
442	0	1	0	0	1	NO
443	0	1	0	0	1	NO
445	1	1	1	1	4	YES
446	0	0	0	1	1	NO
448	0	0	0	0	0	NO
449	0	0	0	0	0	NO
450	1	1	1	1	4	YES
Count stem better:	31	26	22	31		
Percentage:	68.8888889	57.77777778	48.888889	68.88888889		

In looking at those queries which it isn’t better, highlighted in yellow above, we have:

1. cosmic events
2. poaching, wildlife preserves
3. law enforcement, dogs
4. declining birth rates
5. profiling, motorists, police
6. Greek, philosophy, stoicism
7. curbing population growth
8. child labor
9. heroic acts
10. U.S., investment, Africa
11. tourists, violence
12. ship losses
13. antibiotics ineffectiveness

Some of the queries, such as “cosmic events” or “child labour” which are short, and contain basic words, may not be affected at all by the stemming procedure, and would regardless have many matches which are relevant, and therefore this is a case where we would expect stemming to offer little improvement. In many of the other queries, there are many common suffixes or prefixes which may actually provide important information to the query, therefore in stemming we may cause the meaning of the query to change. For example, “motorists” may be stemmed to “motor, or “ineffectiveness” stemmed to “effective” completely altering the meaning of the query.

In those which stemming has a definitive improvement, especially those in which all four metrics were better, they almost all contain one or both of the below:

1. Words in plural form, or with an “s” suffix (e.g. “minorities”, “tires”, “accidents”, “women”)
2. Suffixes that do not add meaning to the word (e.g. “forged”, “salvaging”)

With the most common trait among these queries being case 1.

References

1. Gupta, L. (2023, February 18). *Sort a map by values in java*. HowToDoInJava. https://howtodoinjava.com/java/sort/java-sort-map-by-values/
2. Heavily referenced pseudocode from course content, as well as design suggestions made in campuswire.