CISC689/489-010 Information Retrieval Homework 1 - TA Solutions by Rich Burns

Answer the following questions in your own words. You may discuss them with other students, but you must turn in your own work.

1. (25 points) Suppose we have a collection of $N$ documents and $V$ unique terms. For simplicity, assume that each term appears at most once per document. Answer the following questions assuming Zipf’s law holds.

   (6 points, 2 each).
   
   (a) What proportion of terms appear just once (i.e. in only one document)?
   \[
   \frac{1}{1(1+1)}
   \]
   
   (b) What proportion of terms appear ten times (i.e. in ten documents)?
   \[
   \frac{1}{10(10+1)}
   \]
   
   (c) What proportion of terms appear $D$ times (i.e. in every document)?
   \[
   \frac{1}{D(D+1)}
   \]

Now suppose we are going to index the documents and terms using an inverted index. If $V = 100$ and Zipf’s law is true, roughly how many integers will be stored in the inverted file? What about for $V = 1000$? What is the general expression in terms of $V$?

   (6 points, 2 each). Here is the general expression:

   \[
   V \sum_{n=1}^{N} \frac{1}{n+1}
   \]

   This was a little tricky to derive. The first trick is to think “outside-the-box” in terms of utilizing a series. Notice that \( \sum_{n=1}^{N} \frac{1}{n(n+1)} \) is the % of terms appearing once + % of terms appearing twice + ... + % of terms appearing in all N documents. Following that, \( \sum_{n=1}^{N} \frac{1}{n(n+1)} \) * $V$ is the number of terms appearing once + the number of terms appearing twice + ... + the number of terms appearing in all N documents. Now, each nth document appears in $n$ documents, so it needs to be duplicated $n$ times in the inverted list: \( \sum_{n=1}^{N} \frac{1}{n(n+1)} \) * $V$ * $n$.

   Suppose we used a bit vector index instead of an inverted index. How big (in bytes) would the bit vector index be in terms of $N$ and $V$?

   (6 points).

   \[
   N \times V \times \frac{1}{8} \text{ bits}
   \]

   Suppose we used a signature index with the signature size set to $k = 8192$ bits. How big (in bytes) would the signature file be in terms of $N$ and $V$? What would $k$ need to be to ensure the signature file is smaller than the inverted file?

   (6 points).

   \[
   N \times 8192 \times \frac{1}{8} \text{ bits}
   \]

2. (25 points) This question asks you to compare three different compression algorithms: Elias-$\gamma$ encoding, Elias-$\delta$ encoding, and $v$-byte encoding.
(a) Suppose a word appears at least once in all $N$ documents in a collection. What compression ratio would be achieved using each of the three algorithms to compress document numbers? (Define compression ratio as the total size of the compressed numbers divided by the total size of the uncompressed numbers. Assume an uncompressed number requires 4 bytes of storage.)

(8 points. 2 points for each algorithm, and 2 points for the compression ratio.)

Elias-$\gamma$: $\sum_{n=1}^{N} (2 \log_2 n + 1) \times \frac{1\text{byte}}{8\text{bits}}$

Elias-$\delta$: $\sum_{n=1}^{N} (2 \log \log n + \log n) \times \frac{1\text{byte}}{8\text{bits}}$

$v - \text{byte}$: $\sum_{i=1}^{\log_{128} N} \sum_{j=128^{i-1}}^{128^i-1} i$ Here’s the idea: the outer summation and its iteration of $i$ is going to store that we need $i$ bytes for the current number, and the inner summation is going iterate from 1 to 127, 128 to 128$^2 - 1$, 128$^2$ to 128$^3 - 1$, and so on...

uncompressed: approximately $N \times 4\text{bytes}$

If you assume d-gap encoding of the document numbers, then everything changes. The math will be simpler because all the d-gaps will be 1.

(b) Suppose a word appears in every 10th document, and it appears 5 times in each of those documents (i.e. 5 times in document 1, 5 times in document 11, ...). What compression ratio would be achieved using each of the three algorithms to compress the full inverted list?

(8 points).

Assuming d-gaps here...

Elias-$\gamma$: $\sum_{n=1}^{N/10} (2 \log_2 10 + 1 + 2 \log_2 5 + 1) \times \frac{1\text{byte}}{8\text{bits}}$

Elias-$\delta$: $\sum_{n=1}^{N/10} (2 \log \log 10 + \log 10 + 2 \log \log 5 + \log 5) \times \frac{1\text{byte}}{8\text{bits}}$

$v - \text{byte}$: $\sum_{i=1}^{N/10} *1\text{byte} * 2$ (one for “10” and one for “5”)

uncompressed: approximately $\frac{N}{10} \times 4\text{bytes} * 2$

(c) Suppose a word appears in $n$ documents such that the maximum gap between any two document numbers is 25. Suppose it occurs at least once in each of those documents, but no more than 10 times in any document. What is the minimum compression ratio that could be achieved by each of the three algorithms? What is the maximum compression ratio?

(8 points).

Here, the best-case would be that the word appears in every tenth document only once. The worst-case would be for the word to appear in every document 25 times.

(d) Extra credit (5 points): Is there a way to compress inverted lists such that terms that appear in all $N$ documents require just 1 bit? If so, describe it. Does it provide lossless compression? Can it be decompressed unambiguously? Will it provide better compression ratios on average than the three algorithms above?

Yes – use a leading bit to signal that a term occurs in all documents. This will be lossless if the index does not hold token count per document information. More bits are needed if we want to store this information lossless. Even with these additional bits, the compression ratios will still be better than the above three algorithms.
3. (25 points) This question asks you to compare retrieval models. Suppose we have the query “oil producing nations”, and the three query terms have inverted lists as follows:

\[ I_k \rightarrow (df_k, ctf_k, (doc_i, tf_{ik}), \ldots) \]

\[ \text{oil} \rightarrow (5, 18, (1, 4), (4, 3), (6, 1), (7, 2), (8, 8)) \]

\[ \text{producing} \rightarrow (4, 20, (1, 6), (2, 2), (5, 4), (8, 8)) \]

\[ \text{nations} \rightarrow (3, 11, (1, 1), (3, 2), (8, 8)) \]

Further suppose we have a collection of documents with lengths as follows:

\[ d_1 \rightarrow 498 \]
\[ d_2 \rightarrow 627 \]
\[ d_3 \rightarrow 551 \]
\[ d_4 \rightarrow 648 \]
\[ d_5 \rightarrow 621 \]
\[ d_6 \rightarrow 639 \]
\[ d_7 \rightarrow 566 \]
\[ d_8 \rightarrow 423 \]
\[ d_9 \rightarrow 589 \]
\[ d_{10} \rightarrow 525 \]

and the total number of terms in the corpus is 5687. What are the scores of the 10 documents using each of the following retrieval models:

(a) Vector space model with term weighting \[ w_{ik} = \frac{tf_{ik}}{len_i} \log \frac{N+1}{0.5+df_k} \]

(b) Binary independence model

(c) BM25 model with parameters \[ k_1 = 1.2, b = 0.75, k_3 = 0 \]

(d) Language model with Jelinek-Mercer smoothing parameter \[ \lambda = 0.2 \]

(e) Language model with Dirichlet smoothing parameter \[ \mu = 2000 \]

Explain what makes these models different from each other, focusing especially on how they use term frequency, document frequency, and document length to calculate document scores.

Extra credit (5 points): Pick one of these models and describe how you might modify it to give more weight to terms appearing in the title of a document. You may introduce additional term statistics to support your model, but be sure to define them carefully.

4. (25 points) Now suppose the documents from question 3 have been judged by an assessor as follows:

\[ d_1 \rightarrow \text{not relevant} \]
\[ d_2 \rightarrow \text{not relevant} \]
\[ d_3 \rightarrow \text{relevant} \]
\[ d_4 \rightarrow \text{highly relevant} \]
\[ d_5 \rightarrow \text{not relevant} \]
\[ d_6 \rightarrow \text{relevant} \]
\[ d_7 \rightarrow \text{not relevant} \]
\[ d_8 \rightarrow \text{highly relevant} \]
\[ d_9 \rightarrow \text{not relevant} \]
\[ d_{10} \rightarrow \text{not relevant} \]

Calculate precision at rank 5, recall at rank 5, average precision, and R-precision for each of the five models from question 3. Which performed the best by each measure? Which performed the worst?

If the gain of a nonrelevant document is 0, the gain of a relevant document is 2, and the gain of a highly relevant documents is 4, calculate DCG at rank 10 for each of the models from question 3. Which performed the best?

Extra credit (5 points): Is there a way to define term weights \[ w_{ik} \] for the vector space model such that it performs the best by every evaluation measure? If not, why not? If so, describe it. (Note that you cannot simply assign a number to each term. The weights must be a function of term statistics.)