CS47300 Fall 2020 Assignment 2 Solutions

For this assignment, use the following (short) document corpus. Note that we've already done stemming and stopword removal.

D1 higher term frequenc longer document

D2 repeat term give higher term frequenc

D3 term frequenc import term document

D4 invers document frequenc import term corpu

1 Indexing

1. Give a vector space model representation of documents D1 and D2. For ease of search, the vectors should be alphabetized.

	D1	**D2**
corpu	**0**	**0**
frequenc	**1**	**1**
give	**0**	**1**
higher	**1**	**1**
import	**0**	**0**
invers	**0**	**0**
longer	**1**	**0**
repeat	**0**	**1**
term	**1**	**2**

The most common mistake was forgetting that a document vector in the vector space model includes all terms in the CORPUS, even if they aren't in the document. Which is why we don't actually store document vectors, it is a mathematical model.

2. Build an inverted list index for the corpus. Again, this should be alphabetized. For the weights in the index, assume you are using this index for retrieval with a TFIDF retrieval model, with term frequency (TF) weighted as log2(tfkd)+1 and IDF as log2(N/nk), where tfkd is the term frequency (number of occurrences of the term k in document d), N is the number of documents in the corpus, and nk is the number of documents containing the term k. (Treat log2(0) as -1, so that a term that doesn't exist in the document gets a weight of 0.)

I give here an index that includes the IDF score for each word, and the term count for each document containing that term.

corpu:2	**D4:1**			
document: 0.4	**D1:1**	**D3:1**	**D4:1**	
frequenc: 0	**D1:1**	**D2:1**	**D3:1**	**D4:1**
give:2	**D2:1**			
higher:1	**D1:1**	**D2:1**		
import:1	**D3:1**	**D4:1**		
invers:2	**D4:1**			
longer:2	**D1:1**			
repeat:2	**D2:1**			
term:0	**D1:1**	**D2:2**	**D3:1**	**D4:1**

*I was looking to see that the TF*IDF values were in the index, or that the index at least contained the information needed to calculate them. I was also looking to see evidence that you understood this should be a sparse representation.*

3. Compute the cosine similarity between each document and the query “invers”

   D1 $\frac{1 \cdot 0}{\sqrt{1} \cdot \sqrt{.4^2+1^2+2^2}} = 0$

   D2 $\frac{1 \cdot 0}{\sqrt{1} \cdot \sqrt{2^2+1^2+2^2}} = 0$

   D3 $\frac{1 \cdot 0}{\sqrt{1} \cdot \sqrt{.4^2+1^2}} = 0$

   D4 $\frac{1 \cdot 2}{\sqrt{1} \cdot \sqrt{2^2+.4^2+1^2+2^2}} \approx .7$

4. Compute the cosine similarity between each document and the query “repeat higher cours”

   This depends on how you treat “cours”. Some treated it as a typo and assumed it should have been “corpu”. Below, I assume it is a word not in any of the documents. Either is acceptable.

   D1 $\frac{1 \cdot 0+1 \cdot 1+1 \cdot 0}{\sqrt{1+1+1} \cdot \sqrt{.4^2+1^2+2^2}} \approx .3$

   D2 $\frac{1 \cdot 2+1 \cdot 1+1 \cdot 0}{\sqrt{1+1+1} \cdot \sqrt{2^2+1^2+2^2}} \approx .7$

   D3 $\frac{1 \cdot 0+1 \cdot 0+1 \cdot 0}{\sqrt{1+1+1} \cdot \sqrt{.4^2+1^2}} = 0$

   D4 $\frac{1 \cdot 0+1 \cdot 0+1 \cdot 0}{\sqrt{1+1+1} \cdot \sqrt{2^2+.4^2+1^2+2^2}} = 0$

   The most common error here was to calculate $||D||$ using only the terms in the query and document, not all terms in the document.

5. Why would we want the index alphabetized? What becomes easier to do with the alphabetized index?

This enables quick search (e.g., binary search), although there were some other good reasons, such as merging lists from distributed indexes.

What I really wanted was a clear example of why this was useful for a computer to process it, not human readability.

2 Stemming

Stemming in the above corpus can be both good and bad. Assume document D4 was originally ”inverse document frequency is importance of a term in a corpus”.

1. Give an example of a query that you hope would return D4, and would do so if the query and document were stemmed, but would not return D4 if the query and document were not stemmed.

2. Give an example of a query that you would not want to return D4, but because of stemming, would likely return D4.

   This basically means we have words that are not an exact match, but would stem the same way. We want a query that stemmed matches the words “corpu document frequenc import invers term ”. Although frequenc and term are not likely to matter, since IDF is 0. Something like “important terms inversely related to frequency in documents” would probably not match the unstemmed query (only frequency matches, but it presumably has IDF 0, so the TF*IDF score would still be 0). Assuming important stems to import, terms stems to term, inversely stems to invers, and documents stems to document, the stemmed version is a pretty good match.

3. Prove that stemming may help, and will never hurt, recall. This should be a mathematical proof.

   Recall is #relevant retrieved/#relevant. The #relevant is constant for a given query, so the only change from stemming is #relevant retrieved. If we assume anything with a non-zero score is retrieved (e.g., TF*IDF or Boolean Retrieval), and that if there are matching terms the score is non-zero (again, true for TF*IDF since it is a sum of numbers that are always positive for matching terms and zero for non-matching - and also true for Boolean), then we just need to show that the number of matching terms cannot decrease because of stemming.

Let d be a relevant retrieved document before stemming. Since the score of d is non-zero, it must have at least one matching term. Assume the term $d_i = q_i$ is such a matching term. Since stemming is a deterministic function, $d_i = q_i \rightarrow stem(d_i) = stem(q_i)$. Therefore the stemmed document d' contains at least one matching term $stem(d_i) = stem(q_i)$. Since this holds for all relevant retrieved documents d, the number of relevant retrieved documents cannot decrease. Thus:

$$\begin{aligned} \#relevant\_retrieved' &\geq \#relevant\_retrieved \\ \frac{\#relevant\_retrieved'}{\#relevant} &\geq \frac{\#relevant\_retrieved}{\#relevant} \end{aligned}$$

4. Would you be able to prove that stemming always helps or hurts the F1 score? Either give a proof, or a counterexample showing that such a statement can't be made.

We can't make either statement. Counterexample:

$$F_1 score = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{\frac{relevant\_retrieved}{retrieved} \cdot \frac{relevant\_retrieved}{relevant}}{\frac{relevant\_retrieved}{retrieved} + \frac{relevant\_retrieved}{relevant}}$$

Assume a query important where D3 and D4 are relevant, and (to keep the mathematics clean) assume D3 contains the word important, but D4 would not match because it uses the word document. So $F_1 = \frac{1*0.5}{1+0.5} = \frac{1}{3}$. Now stem to get the original table, and both match, giving $F_1 = \frac{1*1}{1+1} = \frac{1}{2}$. The F_1 score has increased.

Now assume the same scenario, but that only D3 is relevant. Unstemmed, we get $F_1 = \frac{1*1}{1+1} = \frac{1}{2}$. But stemmed, we get $F_1 = \frac{0.5*1}{0.5+1} = \frac{1}{3}$. The F_1 score has decreased.

You can find a discussion and implementations of the Porter stemmer here. Unfortunately, the online version I've used in the past doesn't seem to be working, and the online versions I can find seem to hide cryptominers. For this question, you can provide a description of why you think something would stem in a particular way (e.g., a reasonable rule that would cause the behavior you expect.) Your answers to parts 1 and 2 should not depend on the same words being stemmed differently because of different stemming rules, but should be different queries that have unexpected outcomes under similar stemming rules. Your answers to 3 and 4 should not be dependent on any particular stemming algorithm or rules.

3 Boolean Retrieval

As noted in class, boolean retrieval refers to the idea that a document either matches the query or not. It is not limited to boolean expressions. For example, a boolean retrieval model may support proximity search, e.g., (term frequenc /3) would mean that "term" and "frequenc" occur within 3 words of each other.

1. Which documents in the above corpus would satisfy the query (term frequenc /3)?
   All documents

2. Can you answer this query using only the inverted list index? Explain how, or why you can't.
   No. Because inverted list only shows how many times one term appears in a document. We cannot know the position of terms only looking at inverted list.

3. Is there a better way to answer this query than exhaustively searching the entire corpus? Keep your answer brief - one to two paragraphs is plenty to give the idea.
   You can use a positional index list.
   Example:

Terms	**Documents**
corpu	**D4 (6)**
document	**D1 (5), D3 (5), D4 (2)**
frequenc	**D1 (3), D2 (6), D3 (2), D4 (3)**
give	**D2 (3)**
higher	**D1 (1), D2 (4)**
import	**D3 (3), D4 (4)**
invers	**D4 (1)**
longer	**D1 (4)**
repeat	**D2 (1)**
term	**D1 (2), D2 (2, 5), D3 (1, 4), D4 (5)**

It will just go through the corpus one time and remember the position of each term in each document. When you input a query like this, it just searches from the index list.
Comment: I'm open to any reasonable answers. But if the efficiency of your method was not good, you may lose points.

4 TF/IDF

While this uses the documents given at the beginning, you may or may not be able to use the index you created in Question 1. You'll want to think about why.

Given the query inverse document frequency:

1. Compute the TFIDF scores between the above query and documents D1 and D4, using natural (raw) term frequency count, no document frequency, and cosine similarity.

   TF-IDF using raw term frequency: See Table below

	TF-IDF	
Terms	**D1**	**D4**
corpu	**0**	**1**
document	**1**	**1**
frequenc	**1**	**1**
give	**0**	**0**
higher	**1**	**0**
import	**0**	**1**
invers	**0**	**1**
longer	**1**	**0**
repeat	**0**	**0**
term	**1**	**1**

   $Q = [0, 1, 1, 0, 0, 0, 1, 0, 0, 0]$, $|Q| = \sqrt{3}$, $|D_1| = \sqrt{5}$, $|D_4| = \sqrt{6}$.
   $Sim(Q, D1) = \frac{1+1}{\sqrt{3}*\sqrt{5}} = 0.516$
   $Sim(Q, D4) = \frac{1+1}{\sqrt{3}*\sqrt{6}} = 0.707$

2. Repeat the above TFIDF scores using logarithmic term frequency (1+log(tfkd)) and inverse document frequency (log(N/nd)).

	TF				**IDF**	**TF-IDF**			
Terms	**D1**	**D2**	**D3**	**D4**		**D1**	**D2**	**D3**	**D4**
corpu	**0**	**0**	**0**	**1**	**2**	**0**	**0**	**0**	**2**
document	**1**	**0**	**1**	**1**	**0.415**	**0.415**	**0**	**0.415**	**0.415**
frequenc	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**
give	**0**	**1**	**0**	**0**	**2**	**0**	**2**	**0**	**0**
higher	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**0**
import	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**
invers	**0**	**0**	**0**	**1**	**2**	**0**	**0**	**0**	**2**
longer	**1**	**0**	**0**	**0**	**2**	**2**	**0**	**0**	**0**
repeat	**0**	**1**	**0**	**0**	**2**	**0**	**2**	**0**	**0**
term	**1**	**2**	**2**	**1**	**0**	**0**	**0**	**0**	**0**

3. Give the ranking for the above query for documents D1, D2, D3, and D4. Show your calculations. You may be able to do this without calculating the full TFIDF scores, if so, please describe how you did so.

   Calculation:
   $|D1| = \sqrt{(0.415^2 + 1^2 + 2^2)} = 2.274$
   $|D2| = \sqrt{2^2 + 1^2 + 2^2} = 3$
   $|D3| = \sqrt{(0.415^2 + 1^2)} = 1.083$
   $|D4| = \sqrt{2^2 + 0.415^2 + 1^2 + 2^2} = 3.029$
   sim(Q,D1) $= \frac{1*0.415}{\sqrt{3}*2.274} = 0.105$
   sim(Q,D2) $= \frac{0}{\sqrt{3}*3} = 0$
   sim(Q,D3) $= \frac{1*0.415}{\sqrt{3}*1.083} = 0.221$
   sim(Q,D4) $= \frac{1*0.415+1*2}{\sqrt{3}*3.029} = 0.46$
   Thus, the rank is $D4 > D3 > D1 > D2$

4. What is a problem with using natural (raw) frequency? Explain with an example.

   For term frequency, words that occur many times in a document will dominate the results, even though they aren't that much more important than words that occur only a few times. Using a logarithmic measure mitigates this. For example, if we added "term term term term term term term term" to each document, the cosine similarity between any two documents would be very high, since all vectors would point pretty much in the direction of "term", with only small deviations based on other words.

   If you interpreted raw term frequency to include using IDF=1 (as in part 1), then terms that have little ability to distinguish between documents (e.g., "frequenc" appears in every document) are given equal weight to those that are more effective at distinguishing between documents.

5 Evaluation

Consider the evaluation metrics from class: Precision, Recall, F-1 Score, MAP - Mean Average Precision, and MRR: Mean Reciprocal Rank.

1. Is it possible to build a search system which always achieves high (perfect or near-perfect) recall? Explain why or why not.

   - **Yes. Given a fixed index size, if a search system returns all documents in the index, it will always achieve the higher recall possible (Although not necessarily a perfect recall, if we had, for example, a relevant document that contained none of the terms in the vocabulary).**

2. In what situation a system's MAP performance will be equal to its MRR performance?

   Any of the following situations will lead to the same MRR and MAP performance:

   - **There is only one relevant document.**
   - **There is no relevant document.**
   - **All documents associated with the query are relevant.**
   - **We have a perfect ranking under each query.**

3. What kind of metric is useful to evaluate a question-answering system? Explain your answer.

   - **Need to compare among given metrics: Precision, Recall, F-1 Score, MAP - Mean Average Precision, and MRR: Mean Reciprocal Rank.**
   - **MRR suits well as in the Question-answering system, we often concern with a single answer; thus, the rank of the first correct answer is important.**
   - **Reasonable justification with the metric(s) choice receives points.**