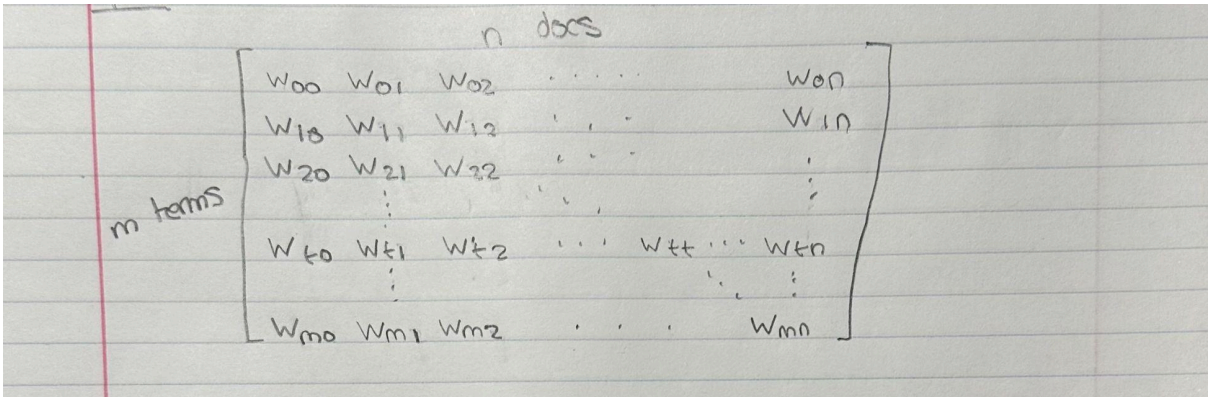
MSCI 541 Homework 5

Thomas Kleinknecht
Watiam: tkleinkn
Student ID: 20883814

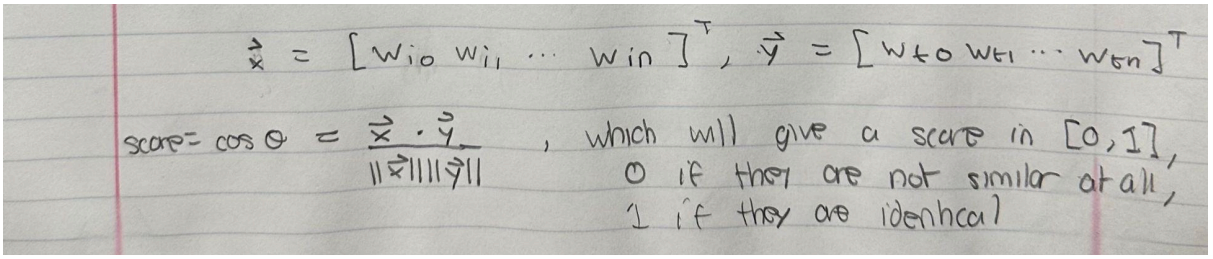
Problem 1

Given the matrix below, which represents what is described in the problem, to compare the similarity of the term t, with all of the other terms 1 to m (not including t), we can use cosine similarity. It was alluded to in the question, as we have seen this before for calculating document similarity.



As seen in the photo below, if we take the transpose of the row vector representing each of the other terms, represented by vector x below, and for term t, represented by vector y below, we can compute the cosine similarity, cos(angle between the vectors), by computing their dot product divided by the product of their norms. As is stated in the below photo, this will give a score in the range of 0 to 1, with 0 being completely different and 1 being identical.

This is given by:
For terms i = 0, 1, 2, … , t - 1, t + 1, t + 2, … , m - 1, m:



In practice, this will measure, for each of the documents these terms have in common, a weight calculated by how much the document is “about” the terms. If the documents are commonly a core focus or important term in similar documents, they must be, by this method, similar terms. Over a very large document collection, this should generate very accurate results, especially for the more niche, very closely linked words.

Problem 2

If I were to use nDCG to measure the quality of a ranked list I would use a gain function with values of 0, 1, and 2, rather than the 5-star system. These values would represent 0 for a

non-relevant recommendation, which the user has zero inclination to watch, a gain of 1 for recommendations which the user could be interested in watching, and a gain of 2 for movies which the user is very interested or definitely would watch. The advantage of using this over a 5-star system is the fact that this system allows us to have a gain of 0 for recommendations which do not provide any value to the user, whereas the 5-star system does not allow for a gain of 0 to be assigned. Also, the decision to limit the gain to three different values was made because it would be difficult, based on a limited view of a recommendation without seeing the movie, to rate it on a 5-point scale. This, however, could be tweaked based on user input if needed.

Problem 3

Computing Query-Biased Summaries:

1. Breaking into “Sentences”
   A very simple scheme was employed for breaking the document up into sentences. First, the headline and pre-text document information is removed, such as length, date, docno, doc id, and section. In the case of no headline being present in the document, all of this is removed in addition to the first 50 characters of the main text as this is used as the headline. The text is then split up on every occurrence of a period (.), question mark (?) and exclamation point (!) into sentences, and stored in a list of sentences, which is then tokenized.

2. Scoring
   a. Metrics used:
      i. l: 2 if it is first sentence, 1 if it is second sentence, 0 otherwise
      ii. C: Equal to total number of query term appearances in the sentence
      iii. d: Equal to the number of distinct query terms in the sentence
      iv. s: Equal to longest contiguous run of query terms in the sentence
   b. Weighted combination:
      Each of these metrics was taken at face value, and summed to generate an overall score for that sentence which was stored in a list of scores for each sentence

3. Output
   The top two highest scoring sentences are outputted, with an ellipses (...) in between the sentences to ensure the user has context in case these sentences do not appear back to back in the document. If, given the sentence breakdown above, only one sentence exists, it is always outputted.

References

1. Heavily referenced pseudocode from course content, as well as design suggestions made in campuswire.