Name: Chung Ting Yang
ID: 31732286

Design

No additional library was used. I added more an addition auxiliary table that stores some of the statistics needed for calculating document scores. Such as average document length and total word count for all documents, scene identifier for each document. This table is serialized to a json file named “documentinfo.json”. Which scoring algorithm to use is passed to the method documentQuery() at run time. Score is calculated for each document and filtered based on which model is used. By filtering, it means things like scores lower than 0 for BM25 are not put into priorityQueue for ranking.

Q6 performance expectation

The result for Q6 should be bad across all models. For BM25, any query term that appears in more than half of the document collection contributes negative score, which could dominate the overall score. Q6 consists of all common terms and its BM25 score for all documents are below 0 which result in no document being considered for ranking.

For likelihood model, since most documents probably contain a lot of terms in the query. Most of them fall in the same score range. Even though the query consist of common terms, it actually has a very specific meaning because how those words are ordered. Likelihood model however, does not take that into account. So a document that actually contains the phrase to be or not to be might get a lower scores than other documents.

Query “setting the scene” performance expectation

This query is likely result in bad performance for BM25 for the same reason. The term “the” is a very common term and the word “scene” shows up in the beginning of every document.

For query likelihood models. I expect the performance to be bad because scene appears in every document at least once and most of them exactly once. So its score contribution will be similar for all documents. The term “setting” appears a lot less than the term “the”, which means the document that contains the highest “the” to document length ratio is going to be rank higher. Which could have very little to nothing to do with “setting” and “scene”

Support phrase queries or other structured query operators

The system needs to implement a query translator that can transform user input queries into a structured query. For example, a query like “to be or not to be” can be transformed in to “to AND be AND or AND not AND to AND be”. The downstream scoring methods can rank documents with all those terms in that order higher. The query translator could also assign words to each query term. For a query like “hope dream sleep”, hope and dream can be interchanged in a lot of context so the user is probably not looking for documents that talk about sleep exclusively.

Methods evaluation

I think both JM and Dirichlet perform at least as good as BM25. There are two reasons for that conclusion. First, Query-Likelihood methods deal with queries with very common terms better than BM25 and that's shown in our experiments above. BM25 can perform poorly if there is no special measure to deal with very common terms in the query. Second, Query-likelihood methods uses probability estimation to compute query term weights, which is likely to outperform untuned BM25 parameters .