CSE272 HW1

Jinghao Shen

May 1, 2022

1 Introduction

This homework works on batch retrieval. First, created a query inverted index from the OHSU88-91 document collection and query.ohsu.1-63 query file. Then, implemented 4 ranking algorithms: boolean, TF, TF*IDF, Relevence feedback. The result is then written to the corresponding log file, following the TREC format. After all the queries in the query set are processed, the trec_eval scoring programis used to determine Precision/Recall values for each ranking algorithm.

Github: `https://github.com/jshen1s1/CSE272/tree/main/HW1`

2 Software Design

The program is written in Python. I used NLTK to build the inverted index. It is an open source python package that enables us to work with natural language and provides handy operations like tokenization and lemmatization with pretrained models.

2.1 Inverted Index

The documents and querys is tokenized using methods in NLTK. To reduce "noise" in the index, I eliminate stopwords and process the token through stemming. By these process, pronouns, vowels, prepositions are removed and words are reduced to their common root-word.

doc_token is structured as {word {document ID}}.
id_word_freq is structured as {document ID {word {word frequency}}}.
query_inverted_index is structured as {query ID {word {document ID}}}.

2.2 Boolean

Boolean ranking algorithm iterate through each query in query_inverted_index. If a document contains a term in that query, the score of the document increases by 1. Then the query is ranked based on the sorted scores.

2.3 TF

TF ranking algorithm iterate through each query in query_inverted_index. The score of each document is calculated based on the frequency of a word in the document. The number of times a word appears in a document divided by the total number of words in the document:

$$TF = \frac{numberofword}{Totalnumberofword} = \frac{n_{i,j}}{\sum_K n_{i,j}}$$

The score of a document is the summation of TF scores of words in that document. Then the query is ranked based on the sorted scores.

2.4 TF*IDF

TF*IDF ranking algorithm is similar to TF ranking algorithm, but include Inverse Data Frequency when calculating scores. Inverse data frequency determines the weight of rare words accross all documents. The log of the number of document divided by the number of document that contains the word w:

$$IDF = log(\frac{numberofdoc}{docwithw}) = log(\frac{N}{df_t})$$

The score from TF is multiplied by IDF. Then the query is ranked based on the sorted scores.

2.5 Relevance Feedback

Rocchio algorithm is used for relevance feedback. Iis aim is to find the optimal query, so to move query vector toward relevant documents and away from non-relevant documents. Rocchio algorithm's formular is:

$$q_m = \alpha * q_o + \beta \frac{1}{|D_r|} \sum_{d_j \in D_r} d_j - \gamma \frac{1}{|D_{nr}|} \sum_{d_j \in D_{nr}} d_j$$

Where: α, β, γ is theweights attatched to each components
We applied Rocchio algorithm by consider the top 10 documents as relevant ones. And set $\alpha = 1, \beta = 0.65, \gamma = 0$.

3 Results

Be referring different measures and scores of 4 ranking algorithm, we can find and infer that:

- By looking at Figure 1, We found that Boolean algorithm performed best among the four algorithms in most measures. Followed by tf*idf and tf. This is expected as tf*idf takes idf into consideration which provides a better result than tf. However, pseudo relevance feedback did not reach the expectation. It may caused by the incorrect implementation of relevance feedback algorithm.

- The Figure 2 shows the relation between recall and precision. As recall increases, the precision decreases. There is a negative relationship between the two.

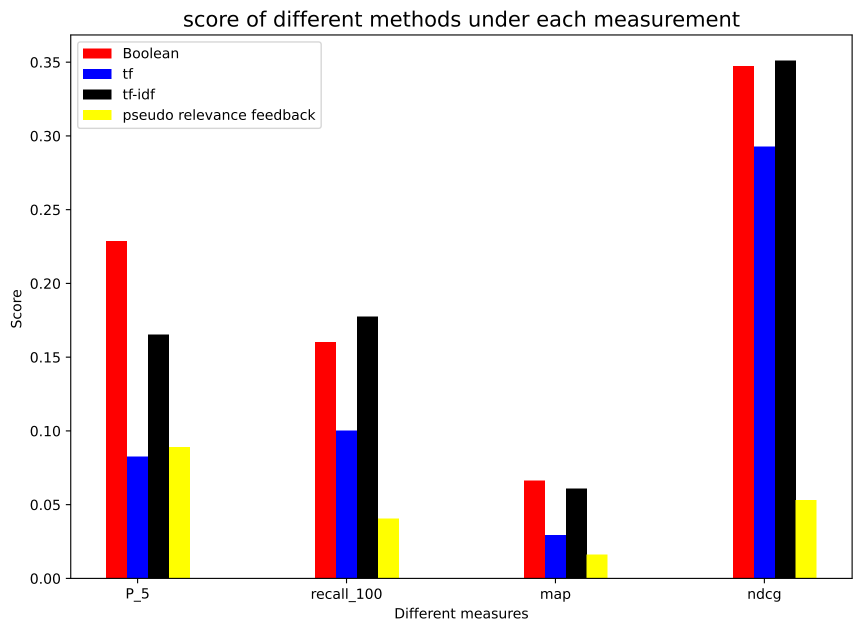

Figure 1: Four ranking algorithms under 4 different measures

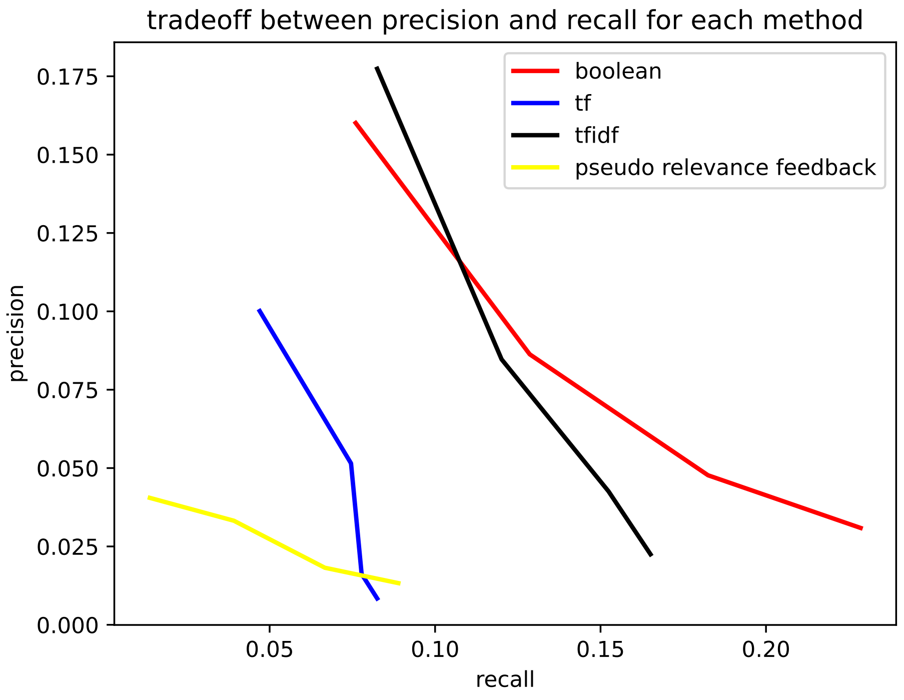

Figure 2: Relation between recall and precision