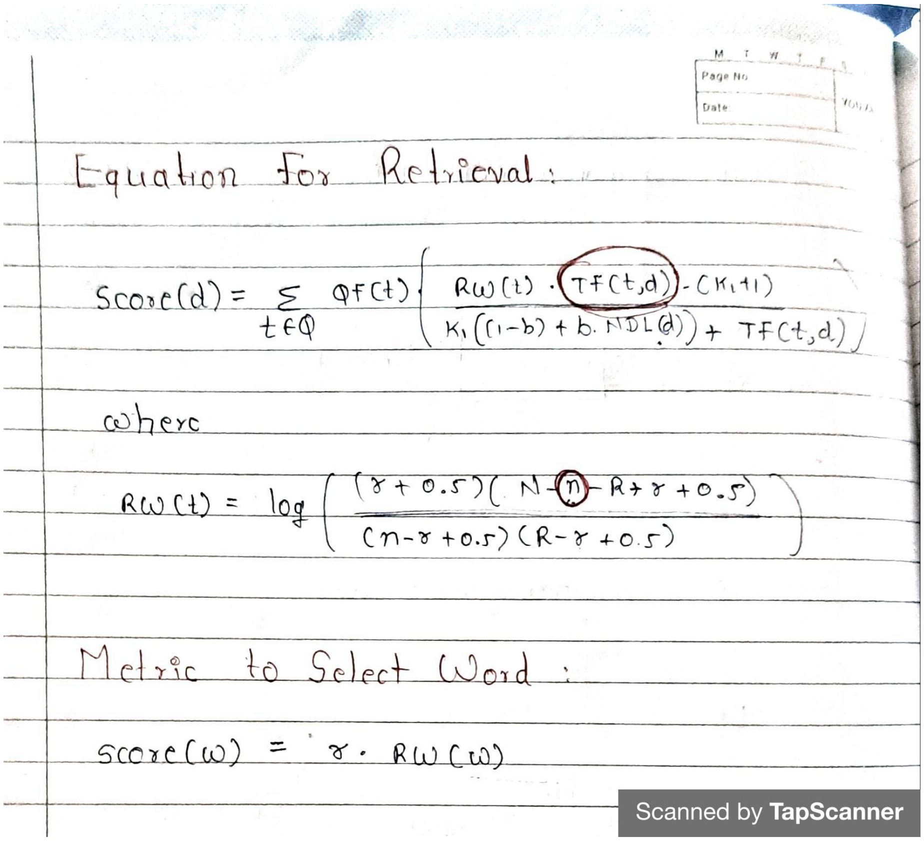
Reranking

Gohil Dwijesh 2017CS50407

November 2020

1 Implementation Details

1.1 Probabilistic Retrieval Query Expansion



Page No

Date

Equation For Retrieval:

$$Score(d) = \sum_{t \in Q} QF(t) \cdot \left(\frac{RW(t) \cdot \boxed{TF(t,d)} \cdot (K_1+1)}{K_1((1-b) + b \cdot NDL(d)) + TF(t,d)} \right)$$

where

$$RW(t) = \log \left(\frac{(r+0.5)(N - \boxed{n} - R + r + 0.5)}{(n-r+0.5)(R-r+0.5)} \right)$$

Metric to Select Word:

$$score(w) = r \cdot RW(w)$$

Figure 1: Equations for probabilistic retrieval model used

- **Data Structure:** Sparse matrix for storing sparse data and numpy matrix for storing dense data. Sets for storing unique words. Dictionary for inverted index.

- **Preprocessing:** Only the stop word removal is applied for query and document words.

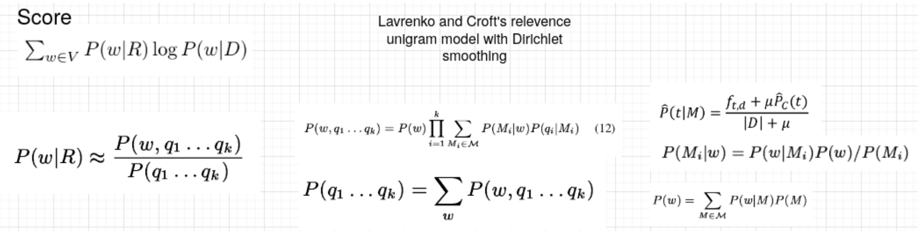
- **Algorithm at High Level:**

  - Store unique words from query and its corresponding 100 relevant docs
  - Get inverted index for words that appear in any of the query or in any of the relevant docs.
  - Calculate score for each of the above word using the formula given in the image. Choose topk words.
  - For all queries, rank all docs using BM25 scoring formula given in the image. Sort documents based on the score and write output in a file.

- **Time Complexity:** O(Q * D) where Q is number queries, D is number of documents in collection.

- **Space Complexity:** O(Q * D).

1.2 Relevance Model based Language Modeling(unigram)



Score

$\sum_{w \in V} P(w|R) \log P(w|D)$

Lavrenko and Croft's relevence unigram model with Dirichlet smoothing

$$P(w|R) \approx \frac{P(w, q_1 \ldots q_k)}{P(q_1 \ldots q_k)}$$

$$P(w, q_1 \ldots q_k) = P(w) \prod_{i=1}^{k} \sum_{M_i \in \mathcal{M}} P(M_i|w) P(q_i|M_i) \quad (12)$$

$$P(q_1 \ldots q_k) = \sum_{w} P(w, q_1 \ldots q_k)$$

$$\hat{P}(t|M) = \frac{f_{t,d} + \mu \hat{P}_C(t)}{|D| + \mu}$$

$$P(M_i|w) = P(w|M_i)P(w)/P(M_i)$$

$$P(w) = \sum_{M \in \mathcal{M}} P(w|M)P(M)$$

Figure 2: Equations from Lavrenko and Croft model for relevance based language modeling

- **Data Structure:** Dictionary data structure is used for storing posting list, and for mapping score for top100 documents for a given query. For other tasks python's primitive data-types have been used.

- **Preprocessing:** For all queries, and documents, all letters are converted to lower case. Stop words have been removed. Words are stemmed.

- **Algorithm at High Level:**

  - Iterate through all queries one by one.
  - For a given query, calculate the KL Divergence score for each document in top100 list. The formula used for calculating score is shown in the figure 2.
  - Sort document based on their score and write the output in a file.

- **Time Complexity** O(Q * D) where Q is number of queries and D is number of documents in the entire collection.

- **Space Complexity** O(1) program stores information about top100 relevant doc at any given time. Which can be treated constant.

1.3 Notes:

- Code for probabilistic retrieval model, and relevance based language modeling in unigram setting works correctly. Bigram features have not been implemented.

- Code for probabilistic retrieval model is both time and memory expensive. So can not run it on large dataset.

- Not able to run code on hpc. So this report does not have empirical results.