Information Retrieval
CS 6200

Professor Nada Naji

FINAL PROJECT
FALL 2016

Aditya Awalkar
Apoorv Anand

Submission Date :-
December 9, 2016

Description

The project's objective is to develop understanding of the core concepts of Information Retrieval.

The project consists of mainly four parts, as following: -

1. Implement four retrieval models namely,
    - TF-IDF (term frequency)
    - Cosine Similarity model
    - BM25
    - Lucene's default retrieval model

2. Implement query expansion using Pseudo Relevance Feedback.

3. Perform stopping and stemming

4. Evaluate and compare above tasks using-
    - MAP (Mean Average Precision)
    - MRR (Mean Reciprocal Rank)
    - P@K (Precision at K)
    - Precision and Recall

Contribution

Aditya Awalkar

- Implemented TF-IDF
- Implemented Cosine-Similarity
- Implemented BM25
- Implemented whole task 3
- Implemented Evaluation
- Implemented Bonus Task of Snippet generation

Apoorv Anand

- Implemented Lucene default retrieval model
- Implemented Task-2
- Implemented Rocchio's feedback Algorithm
- Implemented Lucene with stopping
- Created ReadMe.txt

Implementation of Task-1

TF-IDF Implementation

- Vector space model is used to implement TF-IDF
- Pandas dataframe is used to store the tf-idf values
- Idf values for the query terms are calculated initially
- Tf * idf is done in last step for each document

Cosine Similarity Model

- Tf-IDF values from last run are reused here
- Document magnitude is calculated in a separate file for all the words occurring in the document.
- Dot product is taken for each term with the document
- Finally cosine similarity is calculated by dividing the dot product with the magnitudes of the document and the query

BM25

- BM25 algorithm is used here where a separate function is created for the formula and each query with document is passed to it.
- A document class is created, where all the documents are initialized with their specific fields like word frequency, document length, K, documents name.
- All other fields specific to the query are calculated while looping over the queries.
- Finally the BM25 score for the query is calculated for all documents. Similarly it is calculated for all docuemnts.

Lucene

- Lucene's Simple Analyzer has been used to index the files.
- A tokenized corpus is provided to Lucene to create index.
- Prompts ask for information like location of corpus, index, query file and result file
- The 64 queries provided are then stored in a hash map along with query ids
- Lucene processes these queries one by one and stores the result in Lucene_result.txt file.

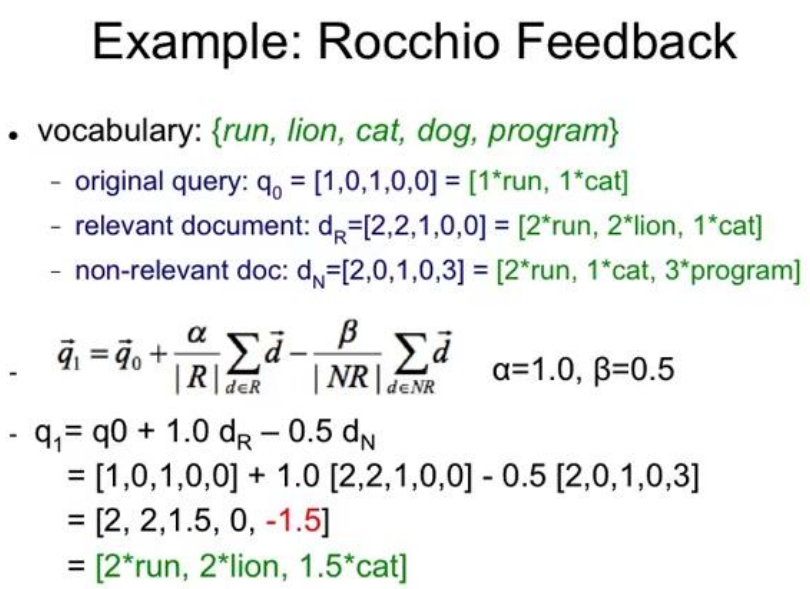
Implementation of Task-2

Query Expansion using Pseudo Relevance (Rocchio Algorithm)

- Run BM25 and take the top 10 results as relevant
- Then use these top 10 documents to expand the query
- Calculate query vector based on the all terms
- Calculate relevant documents vector based on the all terms
- Calculate non-relevant document vector based on the all terms
- Calculate the result using Rocchio formula as follows

  Query_Vector + (Alpha * Relevant Document Vector) – (Beta * Non Relevant Document Vector)

  Where,
  Alpha = 1
  Beta = 0.5



References:-

1. Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze: An Introduction to Information Retrieval, 2009
2. Victor Lavrenko's channel https://www.youtube.com/watch?v=V6u63kTP9Og

Implementation of Task-3

- Stopping is implemented on BM25 by removing the stop words from the query
- Stop words is the list of common words provided which are insignificant
- Stemming is used with the stemmed corpus and the stemmed query
- Again, BM25 is implemented with stem classes which improves the document retrieval

Implementation of Evaluation

Evaluation

- For Evaluation, Precision and Recall is calculated at every step is calculated based on whether the document retrieved is relevant or irrelevant and the total number of relevant documents for the query
- Using the Precision and Recall values, MAP, MRR and P@K metrics is calculated
- The relevance information is used in this case

Snippet Generation

- Since the documents in the corpus are not of passages, and just short title with optional abstract and numbers, we refrained from using significance method for sentences in snippets.
- We employed an easy method where we parsed only the main content of the document which includes the title of the document and optional abstract.
- Next we compared each query term with the newly obtained document.
- Wherever the term was found, it's index is taken and the terms 3 positions before it and 3 positions after it are taken as a sentence if the start or end of file is not reached.
- Such sentences are found for all terms if possible and the snippet is all these sentences combined.
- References: http://stackoverflow.com/questions/28096939/how-to-create-a-google-like-text-snippet-from-string-in-python

Result

BM25
MAP= 0.485664113641
MRR= 0.659375

BM25 with stopping
MAP= 0.499224051945
MRR= 0.647718253968

Tf-Idf
MAP= 0.276416844699
MRR= 0.366234057179

Lucene
MAP= 0.393587948331
MRR= 0.541088002709

Lucene with stopping
MAP= 0.393587948331
MRR= 0.541088002709

Cosine Similarity
MAP= 0.292630805263
MRR= 0.380471042554

BM25 Query Expansion
MAP= 0.494664080224
MRR= 0.679166666667

Conclusion:-

1. BM25 is the best retrieval model among the four models because it considers relevance information. Without relevance information BM25 is like cosine similarity
2. Query expansion gives related word for the query and thus increases the probability of getting more relevant documents and hence increases the precision.
3. Stemming clubs together the words with same stem into one thus reduces the noise and increases the precision
4. Stopping removes the insignificant words and increases precision
5. Snippet generation gives meaningful query result by giving more information about the most important sentences in the document related to the query.

Bibliography

References: -

1. Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze: An Introduction to Information Retrieval, 2009
2. Victor Lavrenko's channel https://www.youtube.com/watch?v=V6u63kTP9Og
3. http://stackoverflow.com/questions/28096939/how-to-create-a-google-like-text-snippet-from-string-in-python
4. Slides taught in CS6220 in fall 2016 at NEU by Professor Nada Naji