INFORMATION RETRIEVAL

PROJECT REPORT

SPRING 2017

INSTRUCTOR: NADA NAJI

TEAM:

AMIT MANGOTRA

NANCY AGARWAL

VAIBHAV KARNAM

Introduction

This project uses Python to build an information retrieval system in two phases. The first phase involves indexing and retrieval and the second phase deals with the evaluation of phase 1 runs.

Contributions

The contributions of the members are as follows:

1) Amit Mangotra:
- Built search engine with BM25 as a retrieval model
- Built search engine with tf-idf as retrieval model
- Documentation of the above

2) Nancy Agrawal:
- Implemented query expansions
- Performed evaluation on the runs of phase 1.
- Documentation of the above

3) Vaibhav Karnam:
- Built search engine with Lucene as a retrieval model
- Implemented indexers and stopping parser
- Documentation of the above

Literature and Resources

1) Retrieval Models:

- **BM25** – The BM25 score is calculated using the formula given below [1][4]

$$\sum_{i \in Q} \log \frac{(r_i + 0.5)/(R - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - R + r_i + 0.5)} \cdot \frac{(k_1 + 1) f_i}{K + f_i} \cdot \frac{(k_2 + 1) q f_i}{k_2 + q f_i}$$

Where summation is over all the terms in the query [2]
- k1 and k2 are constants
- qfi is the frequency of the term within query frequency [2]
- f is the document frequency [2] of the term
- n is the number of documents in the collection indexed by this term
- N is the total number of documents in the collection
- r is the number of relevant documents [2] indexed by this term
- R is the total number of relevant documents [2]

The value of K is calculated by using the given formula [2]

$$K = k_1((1-b) + b \cdot \frac{dl}{avdl})$$

Where,

- dl is the length of the document
- avdl is the average length of the documents in the corpus
- b is a constant

TF-IDF Weighted Sum

Tf-idf stands for *term frequency-inverse document frequency*, and the tf-idf weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus [5].

- **TF: Term Frequency** measures how frequently a term occurs in a document. Since every document is different in length. The term frequency is often divided by the document length (total number of terms in the document) as a way of normalization [8]

  TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

- **IDF: Inverse Document Frequency** measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones. [8]

  IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

LUCENE

provides Java-based indexing and search technology along with well as spellchecking, hit highlighting and advanced analysis/tokenization capabilities. It is an open source project and can be accessed here - https://lucene.apache.org/. Lucene scoring uses a combination of the Vector Space Model (VSM) of Information Retrieval and the Boolean model to determine how relevant a given Document is to a User's query [7]

Query Expansion:

The API used for retrieving synonyms and derivatives of English words is Words API
URL: https://www.wordsapi.com.
The expansion technique used is: **Inflectional and derivational variants. [9]**

1) **Inflectional/Synonyms Variants:**
   Expand the query by adding to the query terms with same (or almost same) meanings. For example, “global” in a query is expanded with synonyms like “worldwide”, “planetary”, “ball-shaped”, “globose”, “spheric”, “world-wide”, “world”, “round”, “spherical”, and “circular”.
   The file inflectionalWords.txt contains all the expanded queries.

2) **Derivational Variants:**
   Expand the query with the derivative variants of a term.
   For example “style” in a query would be expanded with derivatives “styler”, “stylize”, “stylist” and “stylistic”

   This query expansion has been done with the help of two external libraries:

1) **Natural Language Toolkit**:
   URL: http://www.nltk.org/
   Usage: Natural Language Library for processing English words (written in Python)

2) **Unirest**:
   URL: http://unirest.io/
   Usage: HTTP request library

   It is to be noted that stop words in the queries are not expanded. So, for the expansion we consider only meaningful non-stop words. We eliminate these stop words/common words during expansion.

Implementation and Discussion:

Retrieval Models:

TF.IDF

TF.IDF scores are calculated in the program tf-idf.py file. The data structure for inverted index is a dictionary in which index term is the key and the value is a list of tuples of the type (document_id, count). The list of tuples was used so that we can efficiently iterate over the list while ranking the documents for queries. The query file is then parsed to obtain a list of words for each query. This word is then compared against the inverted index. The TF.IDF score is then calculated iteratively over each frequency term of query.

BM25

The data structure used for the inverted index is a dictionary in which index term is the key and value is a list of tuples of the following type - (document_id, count). Here, we use a list of tuples to efficiently iterate over the list while ranking the documents for the queries. The query file is then parsed to obtain a list of words for each query. The word is then compared against the inverted index and the BM25 score is calculated iteratively for each of the frequency term of the query. Then, For each query the number of relevant documents is obtained from cacm.rel file provided and a dictionary is populated in the following format

{queryID:[list of relevant documents]}. This is used to calculate the value of R and ri for BM25.

LUCENE

Lucene is implemented in java by the files IndexFiles.java and SearchFiles.java using the standard library lucene-core-4.7.2.jar. The standard analyser was used which removes a fixed set of stop words. The library imported for this task was lucene-analyzers-common-4.7.2.jar. Lucene-queryparser-4.7.2.jar was used for query parsing.

Query Expansion:

The implementation of query expansion takes the following course:

- The original 64 queries, given in cacm.query.txt file is tokenized by running the file QueryParsing.py file.

- To retrieve the synonyms and the derivative variants of the terms in the tokenized queries, we call the Words API by running the file findSynonymsDerivatives.py. This results in the generation of expanded queries based on synonyms (stored in inflectionalWords.txt) and derivatives (stored in derivativeWords.txt).As detailed, the common words are not considered while query expansion. The common words are considered by running the file findCommonWords.py

- Finally, when we run QueryExpansion.py the expanded query files are generated. Along with the expansion of original queries, we also generated the queries without stop words, and expanded this query set as well.

Query-by-query analysis:

1) Stemmed Query “portabl oper system”

The top most document CACM-3127 was relevant to the query. This shared the same rank in the BM25 retrieval model (with relevance judgement). This document (CACM-3127) contains information about “Thoth, a Portable Real-Time Operating System”. It has all the words in the given query, so it is topically relevant to the query. However, the ranked documents after CACM-3127 are not purely relevant to the given stemmed query. Document CACM-2319 (talks about “Operating System Performance”) is partially relevant. Similarly, document CACM-3068 talks about “A Model for Verification of Data Security in Operating Systems”, CACM-2379 talks about “The Design of the Venus Operating Systems” etc. are partially relevant as they do contain the query terms “oper” and “systems”, but documents like CACM-1591 talks about “A Model for a Multifunctional Teaching System”, CACM-1680 talks about “A General-Purpose Display Processing and Tutorial System” and still exist in the ranking list. The reason for these documents occurring in this ranking list is there are words like “operations”, “operate” etc. present in them and thus, increasing the relevance of a term in a query for document. This is due to “over-stemming” of words.
If the ranking list of BM25 retrieval model is observed for un-stemmed corpus, these documents don’t exist for system searches for exact query terms,using the relevance values given in “cacm.rel” file.

2) Stemmed Query “code optim for space effici”

The top most document in the ranking list for BM25 Retrieval model for stemmed corpus is CACM-2530 which talks about “An Algorithm for extracting Phrases in a Space Optimal fashion” which is not relevant to the query, topically. This very document is not present in the ranking list of BM25 Retrieval Model for un-stemmed corpus. Similarly, documents like CACM-2680, CACM-2863, CACM-3129, and other contain words like “coded”, “optimal”, “space” and hence exist in the ranking list but are not topically relevant. Despite the irrelevancy to the topic, these documents were retrieved. This is a side-effect of “over-stemming” which resulted in fetching non-relevant documents. The system is supposed to retrieve documents which talks about “code optimization”.

3) Stemmed Query “parallel processor in inform retriev”

Comparing the ranking list of BM25 Retrieval Model for stemmed corpus and for un-stemmed corpus, if you look at document (CACM-1262) at second rank for stemmed corpus (which talks about “Procedure-Oriented Language Statements to Facilitate Parallel Processing”), it is quite relevant to the topic of the query. The top most document in the ranking list of BM25 Retrieval Model for stemmed corpus, CACM-1601, talks about “Parallel Numerical Methods for the Solution of Equations” which is not relevant topically to the query. This is again the side-effect of “over-stemming” resulting in retrieval of documents which are irrelevant topically. The documents in the ranking list are supposed to be the ones which talks about “parallel processors in information retrieval”. Similarly, mostly other documents in the ranking list of BM25 Retrieval Model for stemmed corpus are not topically relevant but exist in the ranking list because of side-effect of “over-stemming”, i.e. terms like “parallel”, “processor” occur in the documents but the documents do not talk about the topic of the query. Whereas, in ranking list of BM25 Retrieval Model for un-stemmed, there are documents with pure relevance to the topic of query (e.g. CACM-1811).

Stopping

The List of stop words was given in the file called common_words.txt. These words were added to a list and for the indexer which performs stemming the words present in this list were excluded from the index

System	Retrieval Model	Stemming	Query Expansion	Stopping
1	BM25	No	No	No
2	TF.IDF	No	No	No
3	Lucene	No	No	No
4	BM25	No	Inflectional and Derivational Variants	No
5	TF.IDF	No	Inflectional and Derivational Variants	No
6	BM25	No	No	Yes
7	TF.IDF	No	No	Yes
8	BM25	Yes	No	Yes
9	TF.IDF	Yes	No	Yes

Evaluation/ Results:

cacm.rel that 12 documents numbered,34, 35, 41, 46, 47, 50, 51, 52, 53, 54, 55 and 56 do not have any relevance information, that is they are absent from cacm.rel. Therefore, the evaluation does not take these files into account, because no relevant documents can be found for the queries.

	System 1	System 2	System 3	System 4	System 5	System 6	System 7
MAP	0.5137	0.2739	0.3947	0.5136	0.2596	0.5106	0.3330
MRR	0.7954	0.4889	0.6315	0.7813	0.4554	0.7768	0.5664
P@5	0.4615	0.2461	0.3192	0.4576	0.2461	0.4538	0.3038
P@20	0.2586	0.1548	0.1865	0.2615	0.1644	0.2701	0.1846

Conclusions

The results show that the BM25 retrieval model is better than the other two retrieval models, namely TF.IDF and Lucene. When Comparing the results for baseline runs it was observed that BM25 performed the best for all the evaluation parameters. It has a higher mean average precision(MAP), mean reciprocal rank(MRR), P@5 and P@20 when compared to the other two systems. Therefore, it retrieved more relevant documents earlier and at higher ranks. Also, there were more relevant documents present in top 5 and top 20 ranks. Lucene performs better when compared to TF.IDF.

After performing query expansion using Inflectional and Derivational Variants, the performance of BM25 and TF.IDF were similar to their respective baseline runs. It can be observed that there was an increase in the value of P@20 and a small decrease in the values of other parameters for the systems with query expansion.

For Systems 6 and 7, stopwords were excluded from the index of BM25 and TF.IDF respectively. BM25 without stopwords produced better P@20 scores and small decrease in other scores with respect to BM25 baseline run and Bm25 with query expansion.
TF.IDF produced better results with stopwords removed for all the parameters when compared to TF.IDF baseline run and TF.IDF with query expansion.

Therefore, after comparing all the metrics, BM25 without query expansion and without stopwords removed produced the best results.

Outlook

- Query logs can be used to improve the overall effectiveness of the retrieval system
- Indexer can be improved to store the positional information of the terms in addition to the term frequency
- Quality features such as update count and incoming links can be used in addition topical features of the document
- Page ranks can be calculated for the webpages and used to improve the retrieval system

Bibliography

[1] Course Notes and Slides of CS 6200 Spring 2017, Northeastern University.
[2] Croft, W. Bruce, Donald Metzler, and Trevor Strohman. Search engines: Information retrieval in practice. Vol. 283. Reading: Addison-Wesley, 2010.
[3] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. "Introduction to information retrieval/Christopher D." (2008)
[4] Study of Query Expansion Techniques and Their Application in the Biomedical Information Retrieval A. R. Rivas, E. L. Iglesias, and L. Borrajo. The Scientific World Journal Volume 2014 (2014), Article ID 132158
[5] Using TF-IDF to Determine Word Relevance in Document Queries Juan Ramos 2003
[6] https://docs.python.org
[7] https://lucene.apache.org/
[8] http://www.tfidf.com/
[9] http://www.freepatentsonline.com/6101492.html