PROJECT ON
INFORMATION RETRIEVAL SYSTEMS

CS 6200 INFORMATION RETRIEVAL
FALL 2016

PROF. NADA NAJI

BY
DISHA SULE
SHRADDHA SHAH
TANVI RANADIVE

INTRODUCTION

This report describes an implementation of an information retrieval system with statistical evaluation and comparison of performance based on their retrieval effectiveness.

Retrieval models implemented are:

- BM25
- Cosine Similarity
- TF-IDF
- Retrieval system using Lucene

To increase effectiveness of the retrieval systems, following were implemented:

- Stopping
- Stemming
- Query expansion using Pseudo Relevance Feedback

Effectiveness of the retrieval systems are evaluated using following statistical measures:

- Precision
- Recall
- Mean Average Precision (MAP)
- Mean Reciprocal Rank (MRR)
- Precision at K (P@K)

Implemented snippet generation technique to provide summary of the most relevant documents retrieved for a query.

MEMBER CONTRIBUTION

Disha Sule
Indexing using parsed corpus.
Implemented TF-IDF retrieval model.
Implemented retrieval system using Lucene.
Implemented query expansion using pseudo relevance feedback.
Implemented program to measure effectiveness of the systems.
Snippet generation.
Query and statistical analysis.
Documentation.

Shraddha Shah
Indexing using parsed corpus.
Implemented Cosine similarity retrieval model.
Implemented retrieval system using Lucene.
Cleaned and parsed the corpus.
Parsed the query file to get the queries in the required format.
Implemented stopping and stemming for the retrieval systems.
Implemented program to measure effectiveness of the systems.
Snippet generation.
Query and statistical analysis.
Documentation.

Tanvi Ranadive
Indexing using parsed corpus.
Implemented BM25 retrieval model.
Implemented retrieval system using Lucene.
Implemented query expansion using pseudo relevance feedback.
Implemented program to measure effectiveness of the systems.
Snippet generation.
Query and statistical analysis.
Documentation.

LITERATURE AND RESOURCES

Search engines invoke query expansion to increase the quality of user search results. It is assumed that users do not always formulate search queries using the best terms.

Query expansion is the process of reformulating a seed query to improve retrieval performance in information retrieval operations. In the context of search engines, query expansion involves evaluating a user's input and expanding the search query to match additional documents.

Expanding with terms from the documents that are relevant to the query improves performance by emphasizing terms related to the multiple aspects of the query.

Approaches that could be used for query expansion are:
- Inflectional and/or derivational variants
- Pseudo relevance feedback
- Thesauri, ontologies, etc

The approach used here for query expansion is pseudo relevance feedback using the top retrieved documents. While the other two approaches are commonly used, they are independent of the query and results returned from it. Whereas pseudo relevance feedback adjusts a query relative to the top documents that match a given query. Thus, it gives a dynamic approach which can be adopted for any query using the context of the corpus at hand.

Papers that were reviewed which support the above decision are:

1] A survey on the use of relevance feedback for information access systems
- by Ian Ruthven and Mounia Lalmas

2] http://nlp.stanford.edu/IR-book/html/htmledition/relevance-feedback-and-query-expansion-1.html

3] Positional Relevance Model for Pseudo-Relevance Feedback
- by Yuanhua Lv and ChengXiang Zhai

Third party tools used:
- nltk (Natural language Toolkit) - for tokenizing the corpus
- Lucene (open source library for indexing and retrieval)

IMPLEMENTATION AND DISCUSSION

There are 4 retrieval models used:

1. **Tf-idf**

   Tf-idf is a commonly used term weighting method. It assigns a high weight to a term if it occurs frequently in a document but not very frequently in the entire corpus. On the other hand, it assigns a low weight to a term if it occurs frequently in the corpus which is usually true for stop words.
   The tf-idf score is calculated as the product of the term frequency in the document and the inverse of document frequency (number of documents in which the term occurs).

2. **Vector Space Model**

   The tf-idf values are used to create vector representations of documents with the terms as the axis. The cosine correlation measures the cosine of the angle between the query and the document vectors. The vectors are normalized so that the representation of the vectors are of equal length. Hence cosine correlation will be 1 for two identical vectors and 0 if they are completely dissimilar.

$$Cosine(D_i, Q) = \frac{\sum_{j=1}^{t} d_{ij} \cdot q_j}{\sqrt{\sum_{j=1}^{t} {d_{ij}}^2 \cdot \sum_{j=1}^{t} {q_j}^2}}$$

   The numerator of this measure is the sum of the products of the term weights for the matching query and document terms (known as the *dot product* or inner product). The denominator normalizes this score by dividing by the product of the lengths of the two vectors.

3. **BM25**

   BM25 is a probabilistic retrieval model that is widely used and has been performed successfully across a range of collections especially TREC evaluations. This model focuses on topical relevance of documents and makes an assumption that relevance is binary.
   Each document is given a score which is calculated as:

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

There are some issues related to this formula. If the frequency of the term fi is 0 in the document, the log function will give an error as it is not defined. In this case, the score for that term is not considered for the document. The score for each query term i is calculated for a document. The final bm25 score for that document will be the summation of the scores of all the query terms.
The documents are then sorted according to the highest rank and the top 100 documents are retrieved.

4. Lucene

Lucene is an open source platform which can be used for ranking and searching. This platform has been used with our corpus and query file as input.
Simple analyzer is used in this model for ranking and retrieval with Lucene. Queries have been read from a file and top 100 documents have been retrieved using this model. The top documents for all queries have been written into a file.

5. Query Expansion

The BM25 retrieval model has been selected for query expansion based on its performance and because it takes into account the relevance information, which boosts the results obtained.
Pseudo-relevance feedback has been used for expanding the query to generate top related terms that can be used in place of the query terms for retrieving matching documents.
This method takes the top 5 documents retrieved using the BM25 retrieval model. A weighting scheme using the tf-idf score was used to assign a weight to each term in the top documents. The terms which have the highest weights after applying this method are those terms which are significant for that query in the corpus.
The top 2 words were taken from each of the 5 documents to expand the query.
This expanded query was then fed into the BM25 retrieval model again to get the top 100 documents.

The results using this query expansion method were better than with the BM25 retrieval without query expansion.

6. Snippet generation

Snippet of a document is a summary of the document which is to be displayed to the user. There are two methods of generating snippets. First method is static and does not take query time analysis into consideration. The snippet to be displayed is pre-calculated for each document and the same snippet is displayed irrespective of the query. Second method is dynamic and takes query time evaluation into consideration. Snippets are generated which contain the important words in query for each document.

We have adopted the second method for generating snippets as it takes query into consideration and gives snippets which contain relevant query terms.

Steps taken for snippet generation:

1) For each document, we create a window which consists of n terms. For this corpus, we take n = 10 as the documents are not very large.
2) After computing a score for each window, we slide the window by 1 to get the next window and so on till the end of document is reached. Basically, by sliding the window, we get the window which is most relevant to the query, and can be used to represent the document.
3) For each window, the score is calculated as a combination of tf-idf of each term in the window and a weight for query terms.
   If a term in the document is a query term, the score for that term is boosted by some value. Else, the score will be the tf-idf weight of that term.
4) Thus, the weight for each window is calculated by adding the above weights for each term in it.
5) The windows are ranked by their weights and the window with the highest weight is taken to be displayed as a snippet for that document.
6) Query terms are highlighted in the snippet (In this implementation, the bold tag “ Query Term ” is used to highlight the query terms in snippet)

Here, basically we consider each window as a document and assign it a score. The top one is the one with the highest score and is used to represent the snippet. The snippet for each document is written to a text file in this implementation. Only the top 5 documents are considered for

each of the 64 queries. The same algorithm can be applied to the cumber of documents that are required to be displayed for each query.

Query by Query Analysis:

1) Normal query : performance evaluation and modelling of computer systems

Stemmed query: perform evalu and model of comput system

This query was selected as it has been stemmed to a great extent and is therefore an appropriate candidate to analyze the difference in results obtained by stemming and without stemming.

For the run with normal query, many of the relevant documents were ranked lower than the rank with the stemmed query and stemmed corpus.
Below is a table indicating relevant documents and their ranking:

DocId	Rank without stemming	Rank with stemming
3048	34	1
3089	12	11
1518	32	19
1529	89	40
1533	Not present	76
1572	49	27
3059	59	26
3088	38	34
698	Not present	76

The above table indicates that the ranks of documents were significantly improved when stemming was applied. Since the query could be stemmed quite a lot, the relevant documents and their ranking in the run with stemming of query and corpus proved to be significantly improved.
Some of the relevant documents which were not present at all in the top 100 without stemming, appeared in the top 100 when stemming was performed, as seen from the table above.
Relevant document was found to be at rank #1 with stemming whereas it was at rank #34 in the run without stemming.
More number of relevant documents occurred at earlier ranks when stemming was performed.

2) Normal query: portable operating systems
   Stemmed query: portabl oper system

This query was chosen as it is one of the shortest queries in our collection.

The results of this query were a mixed batch for stemmed and non-stemmed queries and corpus.
Below is the table for results output:

DocId	Rank without stemming	Rank with stemming
2080	29	63
2246	2	2
2629	50	19
3127	1	1

For this query, the number of retrieved documents are equal. However, the ranking of the documents appears to be interesting.
Two of the documents appear at rank #1 and #2 for both runs, with and without stemming.
For the other two ranking of the documents differed with one document appearing at lower rank without stemming as compared to with stemming and other appearing at a higher rank without stemming as compared to with stemming.

Thus, it appears that stemming does not always provide optimal and most relevant results. Sometimes, the query can be too short or too stemmed which can result in good or bad retrieval depending on the relevance considered by the user.

RESULTS

The evaluation of statistical measures for the runs have been consolidated in the below embedded excel workbook:

Evaluations.xlsx

Also Link to the workbook: /IR-Project/Spreadsheets/Evaluations

The top 100 documents for all the queries obtained in the runs have been consolidated in the below embedded excel workbook.

RetrievalSystemRanking.xls

Link to the workbook: /IR-Project/Spreadsheets/RetrievalSystemRanking

The MRR and MAP is consolidated in the below workbook:

MAP_and_MRR.xlsx

Link to the workbook: /IR-Project/Spreadsheets/MAP_and_MRR

The P@K is consolidated in below workbook

P@K.xlsx

Link to the workbook: /IR-Project/Spreadsheets/P@K

CONCLUSIONS AND OUTLOOK

Observation:

1. BM25 performed better than other models in terms of retrieving the relevant documents at higher ranks. BM25 also takes into account the relevance information that is not accounted for in the other models considered in this project.
   To demonstrate with an example with Query1 for Top 100 documents

Relevant DocId	BM25 Rank	VSM Rank
1410	1	Not Present
1572	12	Not Present
1605	2	81

   BM25 at times performs better than Lucene
   To demonstrate with an example with Query2 for Top 100 documents. There are only 3 relevant documents for Query2 and BM25 retrieves it at top 3 positions while Lucene has non relevant documents at position 3 and 4.

Relevant DocId	BM25 Rank	Lucene Rank
2434	2	1
2863	3	5
3078	1	2

   Based on the above observations we chose BM25 for query expansion.

2. Query Expansion boosted the performance of BM25 and increased the MAP and MRR. Query Expansion with pseudo relevance feedback enhances the query by adding relevant terms in the query that are present in the corpus.
   Example:

   Query: *intermediate languages used in construction of multi targeted compilers tcoll*
   Expanded query: *intermediate languages used in construction of multi targeted compilers tcoll nonsingular matrices neliac the machines automata science undergraduate rrp grammars*.

   In the above example we can see that the terms automata, machines, grammers are relevant to the query term compiler and hence effectively enhance the query.

Outlook:

Since we have observed that query expansion usually increases the performance, incorporating smarter ways of query expansion will give us optimal results. So far pseudo relevance expands the query using terms from the corpus of the top k documents.
The query can be further enriched by adding synonyms of the query word that may be present in the document.

We envision to incorporate this feature in our project by using NLTK's WordNet.
WordNet® is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words.
Example:
For example: if our query is 'dangerous fish'
And a relevant document in the corpus is about Sharks
Wordnet would be able to generate Synsets that would lead to Shark since it uses a inter linked network of meaningful words.

CITATIONS AND BIBLIOGRAPHY

1. https://wordnet.princeton.edu/
2. http://times.cs.uiuc.edu/czhai/pub/sigir10-prm.pdf
3. http://www.dcs.gla.ac.uk/~mounia/Papers/ker_ruthven_lalmas.pdf
4. http://nlp.stanford.edu/IR-book/html/htmledition/relevance-feedback-and-query-expansion-1.html
5. https://www.hindawi.com/journals/tswj/2014/132158/
6. http://nlp.stanford.edu/IR-book/html/htmledition/results-snippets-1.html
7. http://ad-publications.informatik.uni-freiburg.de/TOIS_snippets_BC_2014.pdf
8. Search Engines: Information Retrieval in Practice- Croft, Strohman, Metzler
9. www.wikipedia.org
10. http://scholar.colorado.edu/cgi/viewcontent.cgi?article=1044&context=csci_gradetds