KOÇ UNIVERSITY

COMP 540 Project 1

Building An IR System

Handan Kılınç

13.12.2013

1. Project Description

This project requires that we use a server to build variations on a simple search engine. For each variation we run and evaluate a number of queries. The server that gives corpus statistics and queries that will be evaluated are provided.

The server has four databases according to if they use stemming or stop-words. We use database 2 and 3 to compare results.

Database 2 (No Stem and Stop)

Number of docs = 84678, Number of terms = 244011877, Number of Unique Terms = 207224, Average Document Length = 288

Database 3: (Stem and Stop)

Number of docs = 84678, Number of terms = 244011877, Number of Unique Terms = 166054, Average Document Length = 288

2. Steps of the Construction

Step 1: Query Processing

1. All characters are changed to lower-case.
2. All non-numeric and non-letter characters are removed from the query.
3. Stop words in the "stoplist.txt" are cleaned from the query.
4. The first words like "Documents will indentify" etc. are removed.

Note that we didn't do stemming because the corpus gives the same statistics for stemmed and unstemmed words. Therefore, even we would do stemming it does not affect the result.

Step 2: Calculate Model scores

1. For each query term following information is received from the server:
   - Number of times the term occurs in the collection (ctf)
   - Total number of documents containing the term (df)
2. For each document that contains query terms following information is received from the server:
   - Document id
   - Document length
   - Frequency of the query term in the document (tf)
3. Create document list that contain information in step 2 for each document. This list has unique documents and each of them has a query term list with their tf information.
4. Scores of models are calculated for each document.
5. Documents are sorted with descending order according to their scores. (There are 5 sorted document lists and each of them sorted according to one model)

Step 3: Evaluation of results

1. Elements in the sorted document lists are printed with the format that is specified in the project description.
   query-number Q0 *document-id rank score* okapitf
   query-number Q0 *document-id rank score* okapitfIdf
   query-number Q0 *document-id rank score* laplace
   query-number Q0 *document-id rank score* jelinek
   query-number Q0 *document-id rank score* BM25
2. All queries are evaluated with "*trec_eval*" program and NIST qrel file "*qrels.adhoc.51-100.AP89*".

3. Models in the Project

Vector Space Model (Okapitf): Okapitf is calculated for each document:

$$\text{Okapitf} = \frac{tf}{tf + 0.5 + {doclen}/{avgdoclen}}$$

Then the score for vector space model is calculated with dot product of okapitf scores of document term list and okapitf scores of query term list.

Note that when we are calculating okapitf for terms in the query, we substitute query length (number of words in the query) instead of *doclen* and average query length (number of word in 25 query divided by 25) instead of *avgdoclen*.

Vector Space Model (Okapitf*Idf): The idf is calculated for each document. (N is number of documents).

$$Idf = \log\left(\frac{N}{df}\right)$$

Then for each term in the document okapitf * idf is calculated. Then the score for vector space model is calculated with dot product of okapitf*idf scores of document term list and okapitf*idf scores of query term list.

Language Model (Laplace Smoothing): The following formula is used to calculate Laplace smoothing score for each document.

$$\sum_{i \in Q} \log\left(\frac{tf_i + 1}{doclen + C}\right)$$

C = $number\ of\ \ terms\ in\ corpus$

Language Model (Jelinek-Mercer Smoothing): The following formula is used to calculate Jelinek-Mercer smoothing score for each document.

$$\sum_{i \in Q} \log\left(\lambda . \frac{tf_i}{doclen} + (1 - \lambda)\frac{ctf_i}{\#\ terms\ in\ corpus}\right)$$

We take $\lambda = 0.8$.

Poisson Model (BM25): The following formula is used to calculate BM25 score for each document.

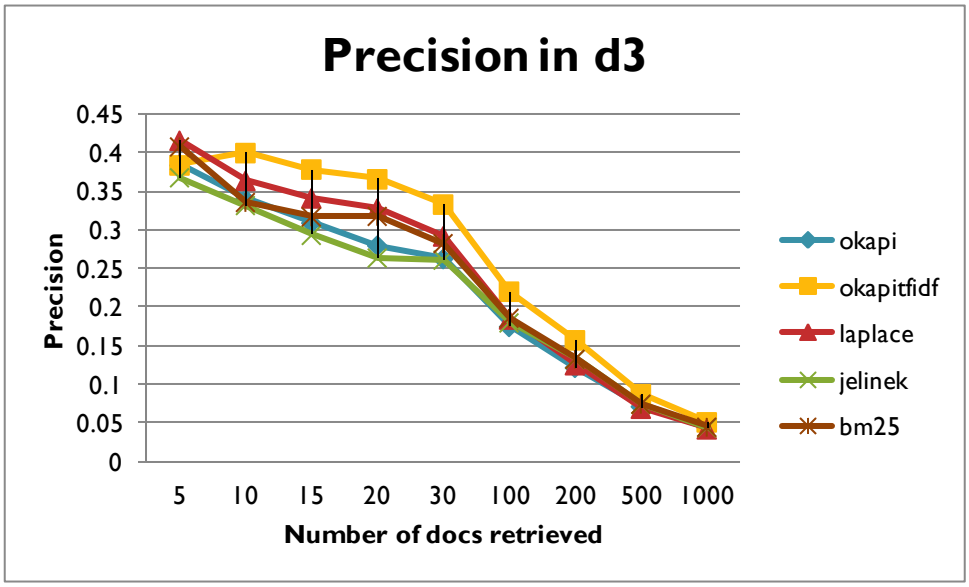
$$\sum_{i \in Q} \log\left(\frac{C}{df_i}\right) . \frac{(k_1 + 1)tf_i}{K + tf_i} . \frac{(k_2 + 1).qtf_i}{k_2 + qtf_i}$$

Where $k_1 = 1.2$, $k_2 = 100$, $K = k_1(1 - b + b . \frac{doclen}{avgdoclen})$, $b = 0.75$.

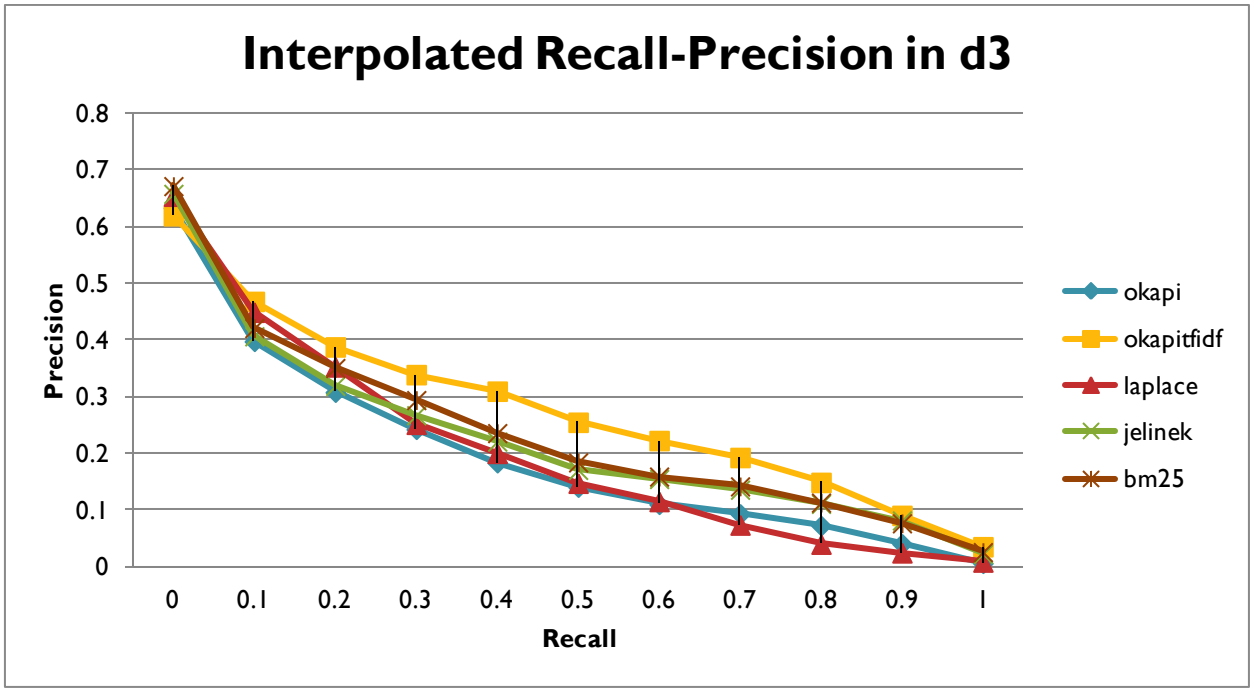
4. Results of Retrieved Documents

WE GET FOLLOWING RESULTS WITH USING DATABASE 3.

Model Name	**Average Precision**	**R-Precision**	**Precision at 10**	**Precision at 30**
Okapitf	0.1771	0.2143	**0.34**	0.2627
Okapitf*IDF	0.2627	0.2931	0.4	0.3333
Laplace	0.185	0.2214	0.364	0.292
Jelinek-Mercer	0.2091	0.2316	0.332	0.2613
BM25	0.2197	0.2461	0.336	0.2827



Graphic 1: Precision in d3 for all models



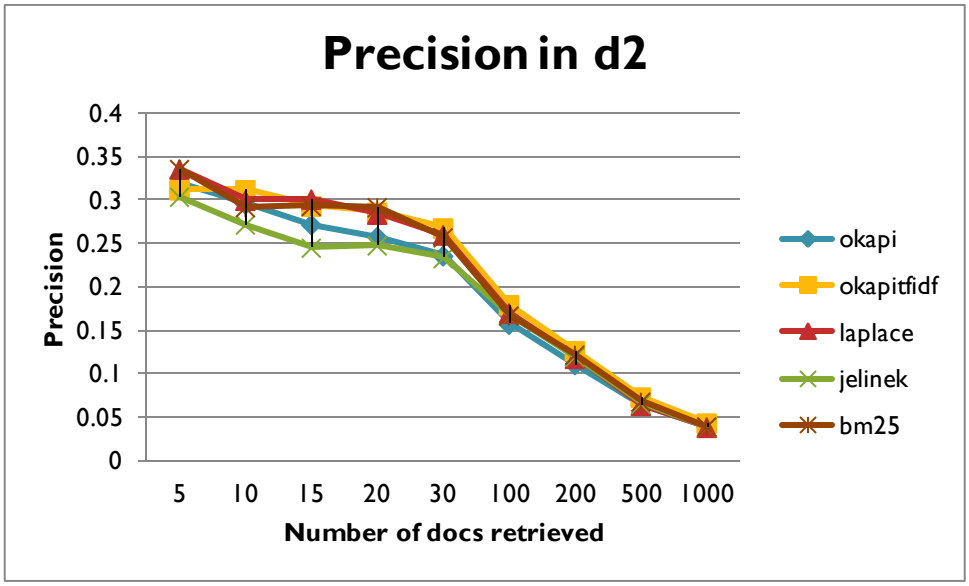
Graphic 2: Recall-Precision Graphic in d3 for all models

- Okapitf*IDF model has the highest average precision and then BM25 model is the second highest.
- As you see in Graphic 1, the precision at Laplace and BM25 model are higher than precision at Okapitf*IDF until the retrieved doc number is around 8. It shows that number of relevant retrieved document is more at first 8 documents in BM25 and Laplace smoothing. After retrieved 8 documents, Okapitf*IDF precision is always more than other models.
  Graphic 1 shows that Okapitf*IDF retrieved more relevant documents than other models.
- We can see the similar result in Graphic 2 which shows that BM25 gives more relevant documents in the beginning.
- Okapi and Laplace gives less relevant retrieved documents than other models according to Graphic 2.

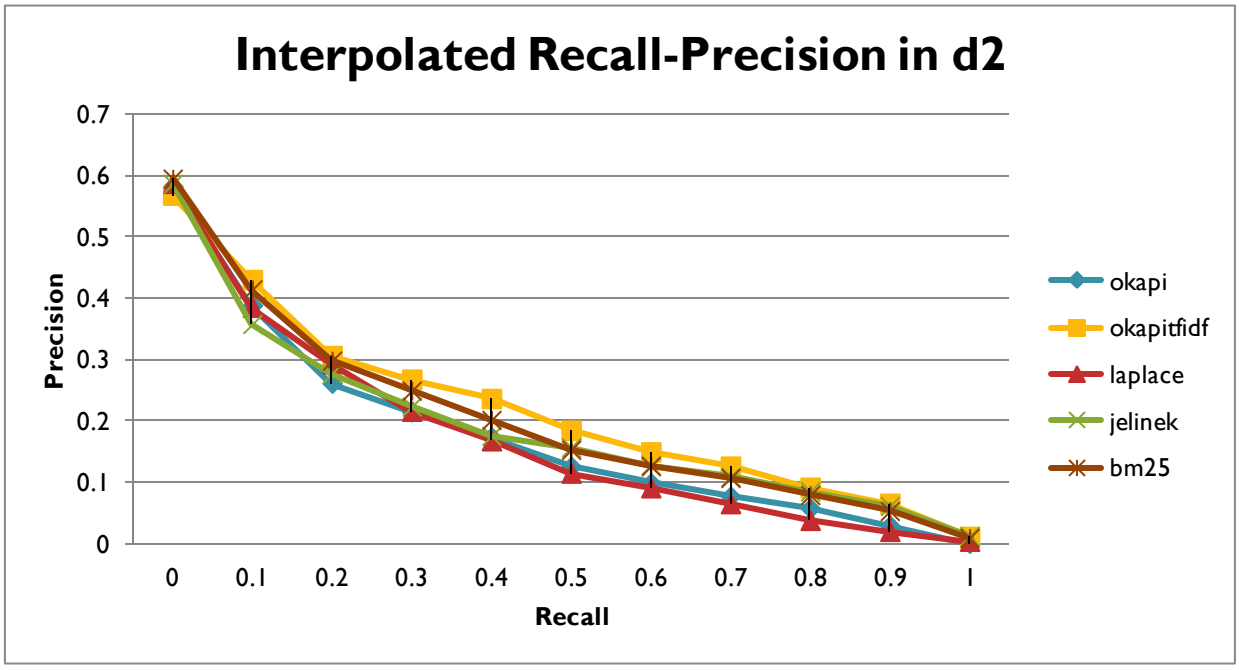
Result: If we want to retrieve more relevant document, it is better to use Okapitf*IDF. However, if we want to retrieve relevant documents in the beginning BM25 is better.

WE GET FOLLOWING RESULTS WITH USING DATABASE 2

Model Name	**Average Precision**	**R-Precision**	**Precision at 10**	**Precision at 30**
Okapitf	0.1599	0.1966	0.296	0.236
Okapitf*IDF	0.2045	0.2243	0.312	0.268
Laplace	0.1571	0.1992	0.3	0.26
Jelinek-Mercer	0.177	0.2072	0.272	0.2333
BM25	0.1869	0.2142	0.292	0.2573



Graphic 3: Precision in d2 for all models



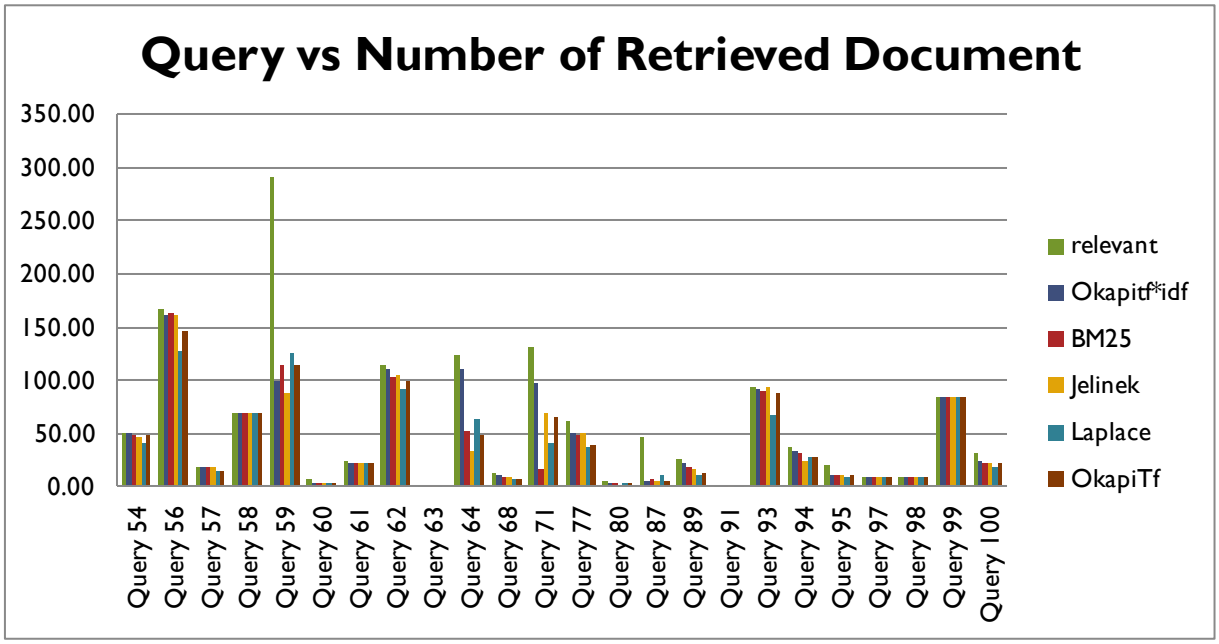
Graphic 4: Recall-Precision Graphic in d2 for all models

- In database 2, we get lower average values than database 3. The reason for this that database 3 does not use stemming. Since ctf values of stemmed and not stemmed words are different, there is decrease in average precision and other values in the table.
- As in database 3, Okapitf*idf gets the highest average precision value and the second one is BM25.
- Graphic 3 shows that in first 8 documents laplace and BM25 gives more relevant documents.
- Graphic 4 shows that okapitf*IDF retrieved more relevant document than others. However the difference is less than database 3. BM25 has almost same number of relevant documents.

Result: Stemming is helpful to retrieve more relevant documents.

5. Query Examination

Following graph shows that number of retrieved documents in database 3 for each model with each query. It shows also real number of relevant documents.



Graphic 5: Number of retrieved documents for each model in database 3

As you see, there is big difference in number of retrieved relevant document and relevant document for query 59, 71 and 87. Therefore, we changed query words without changing the meaning to see if it is related with the words that are used in query. These queries are below:

Query59: *a type of weather event which has directly caused at least one fatality in some location.*

Query71*: incursions by land, air, or water into the border area of one country by military forces of a second country or a guerrilla group based in a second country.*

Query87: *on current criminal actions against officers of a failed us financial institution.*

In query 59 the word "fatality" looks like rarely used, therefore we changed it with death which is more popular word.

In query 71 the words "air", "land" and "water" are not necessary because it incursions can be done only by them. In addition "second country" phrased is removed in the end of sentence.

In query 87, we used "American banks" instead of "us fanatical institution" because bank looks like more popular.

According to change of the queries, following change is seen:

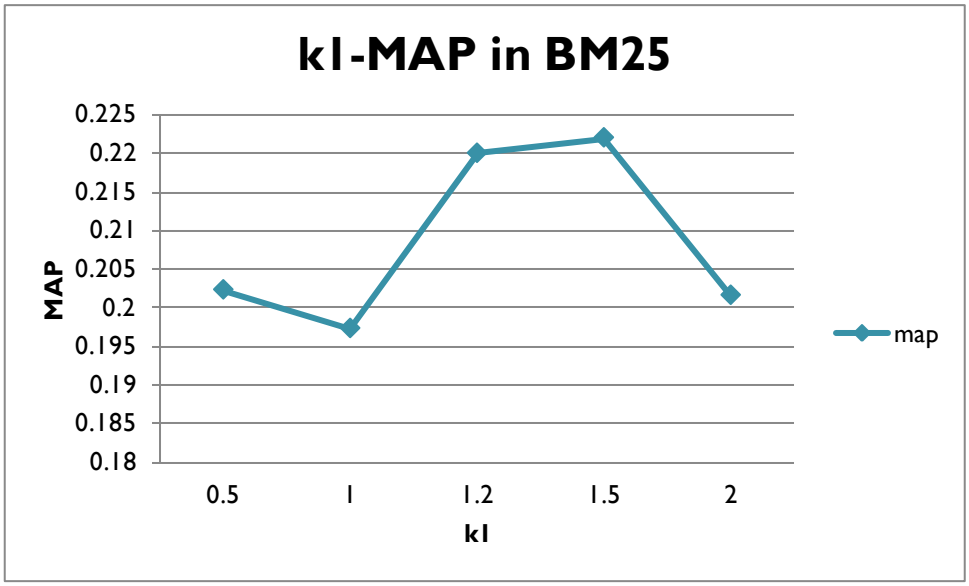
Model Name	Before Query Changed	After Query Changed

	Q59	Q71	Q87	Q59	Q71	Q87
Okapitf	115	65	6	128	77	17
Okapitf*IDF	100	97	5	122	106	15
Laplace	125	42	11	131	60	18
Jelinek-Mercer	89	69	6	109	86	11
BM25	115	17	7	126	40	15

As you seen numbers of retrieved documents are increased for all models.

Result: There can be some process that finds popular similar meaning of a query term so that more relevant documents are retrieved.

6. Different k1 values for BM25 in Database 3



Graphic 6: Different k1 values for MAP of BM25 in d3

Result: The k1 values between 1.2 and 1.5 are best values for BM25 in database 3.