Comp 540: Information Retrieval **Fall 2013**

Project 1 — December 17, 2013

Osman Başkaya Lecturer: Emine Yilmaz

1.1 Introduction

In this project, we are asked to implement various procedures in the IR domain, construct search engines with respective methods and evaluate their performance. In order to achieve this goal, we are given ready-to-use index database, named Lemur Project. By this interface, we can query some statistics such as term frequency, corpus frequency and so on. Addition to these materials, Lemur project has 4 different index databases, according to stem and stopword informations.

Algorithms We have five variations of a retrieval system that needs to be implemented.

1. **Okapi-TF**
2. **Okapi-TF * IDF**
3. **LM with Laplace Smoothing**
4. **LM with Jelinek-Mercer Smoothing!**
5. **BM25**

For the bare minimum, one needs to implement these variations and evaluate their performance on database 3. Database 3 is a stemmed and contains no stopping words.

This paper is organized as follows. Section 1.2 presents the scores of aforementioned algorithms on the database 3. Section 1.3 explains what additional experiments have been done. Section 1.4 concludes the experiments and explains what I have learned.

1.2 Results on Database 3

This section contains the performance scores of 5 different variations in IR on the default database, database 3. Table 1.1 summarizes the performance.

1.3 Additional Experiments

Addition to the requirements, I have done the followings.

1. Results on other databases: Lemure project provides four index databases according to stemming and stopwording choices. I made experiments on all these databases.

Method	Average Precision	Precision on 10 Docs	Precision on 30 Docs.
OKTF	.1383	.2440	.2093
OKTF * IDF	.1606	.1920	.1787
Laplace	.1343	.2760	.2147
Jelinek-Mercer	.1477	.2320	.1947
BM25	.1878	.2760	.2373

Table 1.1. Scores on Database 3 (No query pre-processing has been done)

Method	Database 0			Database 1			Database 2			Database 3		
	AP	**P10**	**P30**	**AP**	**P10**	**P30**	**AP**	**P10**	**P30**	**AP**	**P10**	**P30**
RTF	.0389	.1400	.0947	.0439	.1560	.1120	.1135	.2320	.1907	.1089	.2440	0.2000
OKTF	.0489	.1560	.0920	.0624	.1720	.1160	.1035	.2000	.1653	.1383	.2440	.2093
OKTF * IDF	.0847	.1080	.0960	.1648	.1960	.1853	.828	.1080	.08393	.1606	.1920	.1787
Laplace	.0201	.0560	.0640	.0380	.1240	.0760	.0966	.1960	.1640	.1343	.2760	.2147
Jelinek-Mercer	.0824	.1480	.1360	.1092	.1800	.1667	.1117	.1920	.1493	.1477	.2320	.1947
BM25	.0763	.0960	.0933	.1055	.1280	.1173	.1345	.1920	.1827	.1878	.2760	.2373

Table 1.2. Scores on all database (No query pre-processing has been done. RTF is Robertson's TF)

2. Query preprocessing: Given queries have irrelevant words such as "Document will cite, report, describe discuss identify". These are information for the annotators of the search engine, not for the search engine itself. Thus, I removed those irrelevant words and rerun the all experiments. Punctuation marks are removed, all letters are converted into lowercase equivalents.

3. Robertson's TF: I made some research on Robertson's TF and I found this. In the referenced document, Robertson's TF explained as following: $RTF = TF/(TF + k)$, where $k = 1$. I included this TF method into the evaluation.

4. Figures: In order to understand the performance better, I have drawn some figures.

The remaining part of the section is as follows. Subsection 1.3.1 presents the results on other databases included Robertson's TF. Subsection 1.3.2 demonstrates the performance differences between unprocessed query results (results in Subsection 1.3.1) and the first 3 terms removed query results and additional preprocessing (lowercase, punctuation removing). Finally, in Subsection 1.3.3, I have drawn figures for all the tables.

1.3.1 Results on other databases

I made experiments on the other databases. Please check the figures.

Methods	Database 0			Database 1			Database 2			Database 3		
	AP	**P10**	**P30**	**AP**	**P10**	**P30**	**AP**	**P10**	**P30**	**AP**	**P10**	**P30**
RTF	.0621	.2160	.1333	.0743	.2120	.1480	.1401	.2760	.2200	.1445	.2760	0.2213
OKTF	.0686	.2280	.1467	.0883	.2280	.1787	.1426	.2760	.2200	.1593	.2920	.2360
OKTF * IDF	.1801	.2960	.2587	.2230	.3600	.2947	.1866	.2960	.2533	.2293	.3480	.2960
Laplace	.0358	.1520	.1027	.0507	.1520	.0987	.1278	.2400	.2133	.1514	.3160	.2387
Jelinek-Mercer	.1280	.2360	.1973	.1580	.2600	.2080	.1620	.2560	.2227	.1821	.2680	.2427
BM25	.1115	.1680	.1427	.1277	.1760	.1560	.1877	.3000	.2480	.2239	.3000	.2880

Table 1.3. Scores on all database after query processing. Punctuation removing, lowercase mapping, irrelevant word cleaning has been done as pre-processing. RTF is Robertson's TF)

1.3.2 Query Preprocessing

I had run couple of experiments adjusting query pre-processing steps. I had done (1) lowercase mapping, (2) irrelevant word cleaning, (3) punctuation removing. These all affect the scores. Because of the overwhelmingly many scores, I have only presented none and all-in-all versions. Table 1.1 shows the scores on database 3 only. Table 1.2 demonstrates the raw scores against all database types. Raw scores means that I do not make (1) irrelevant word cleaning and (2) lowercase mapping. Note that I *do* stemming and stop word removing for these experiments according to the database type though. Table 1.3 shows the results after full query pre-processing. As you see, there are significant improvement in performance of all methods.

1.3.3 Figures

There are four figures for the raw results (the results in which there is no further query preprocessing has been done. Please see Subsection 1.3.2), four figures for the results in which all queries have been done further pre-processing steps. Figure 1-4 concludes the raw results, 5-8 demonstrates the query-processed counterpart. I drew one figure for each database and for each method (pre-processed/no preprocessing step). I need to mention the name convention in figures. *vsm-database-3* means that OKAPI*IDF method and *robertson-database-3* is the result of the Robertson's TF method. The rest is straightforward.

1.4 Conclusion

Although document is a bit confusing, it is good to have some hands-on experiences on Information Retrieval. I have seen that parameters are very important and some systems are not robust; if you do some small changes in formula, you will obtain very different results. Addition to all of these, stemming work well if you consider database2 and database3. Performance of all methods improved when we use the databases that stemmed. Similarly, if you compare the scores between database0 and database1, stopping word removing affects the performance positively. Again, all scores are improved. It is worth noting that, OKTF * IDF, Jelinek-Mercer. BM25 methods have achieved good results even other methods totally messed up. Please see database0 in Table 1.3. IDF (or similar) parts of these methods give

them robustness and they are endurable on stopwords.

Another issue is the query sets. **Queries were for annotators not for the search engines!** So, this kind of project should give great deal of attention for this kind of important issues. We should not implement the project without thinking. We need to get some lessons from projects. That should be the aim. If you give these kind of queries for a bare minimum system, it demotivates the people since it is completely meaningless to evaluate the performance such a query, for instance, *"Document will discuss how MCI has been doing since the Bell System breakup"*. I have written considerable amount of code and gave significant amount of time for this project. It is much better for us to create a modest search engine with important details as a bare minimum. Documentation of the project is also not well and can arise a bit of controversy. Project is not well-prepared and it should be revised. I did not learn enough when I compared the time I gave.

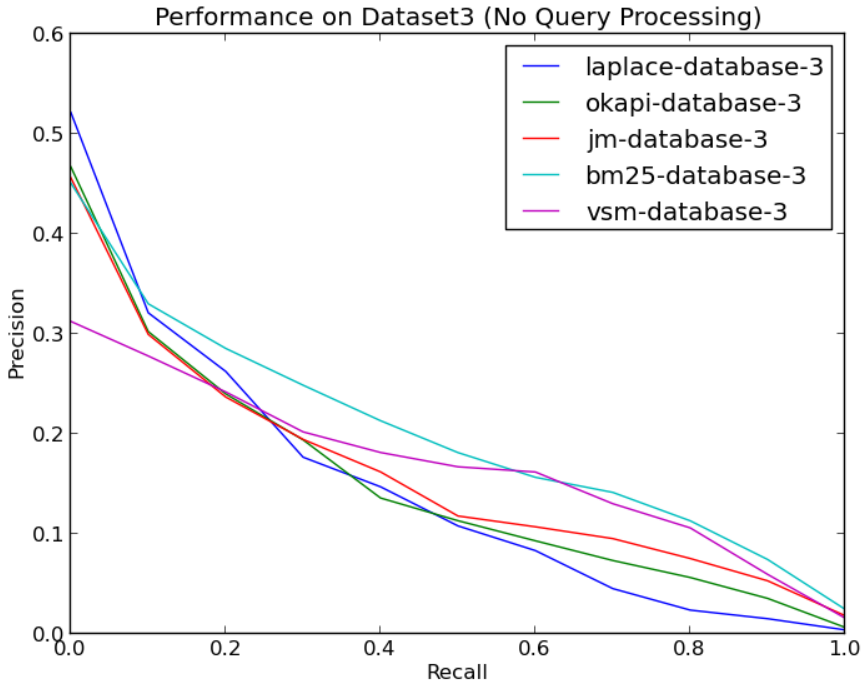

Figure 1.1. Results on Database 3 (No query processing has been done)

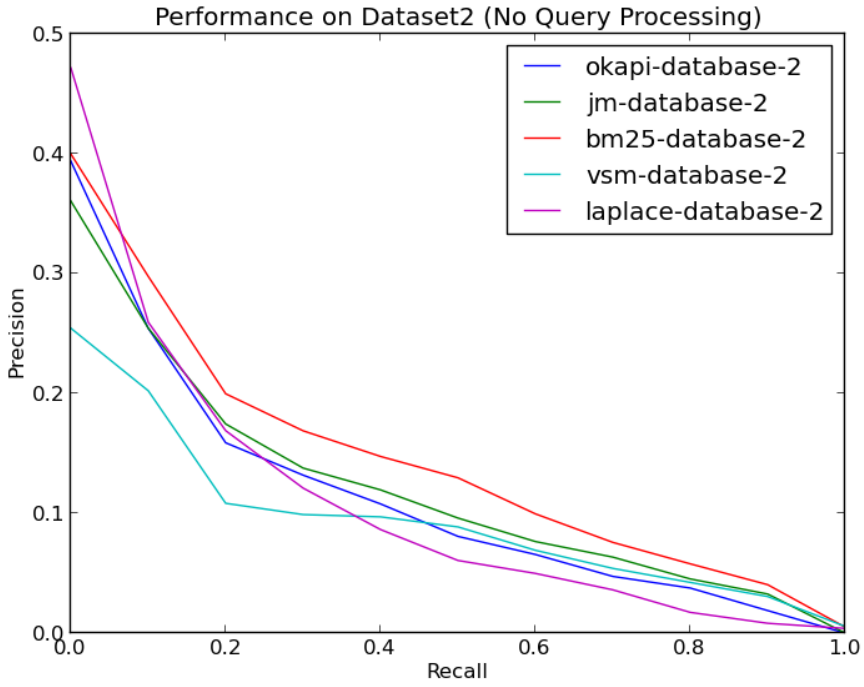

Figure 1.2. Results on Database 2 (No query processing has been done)

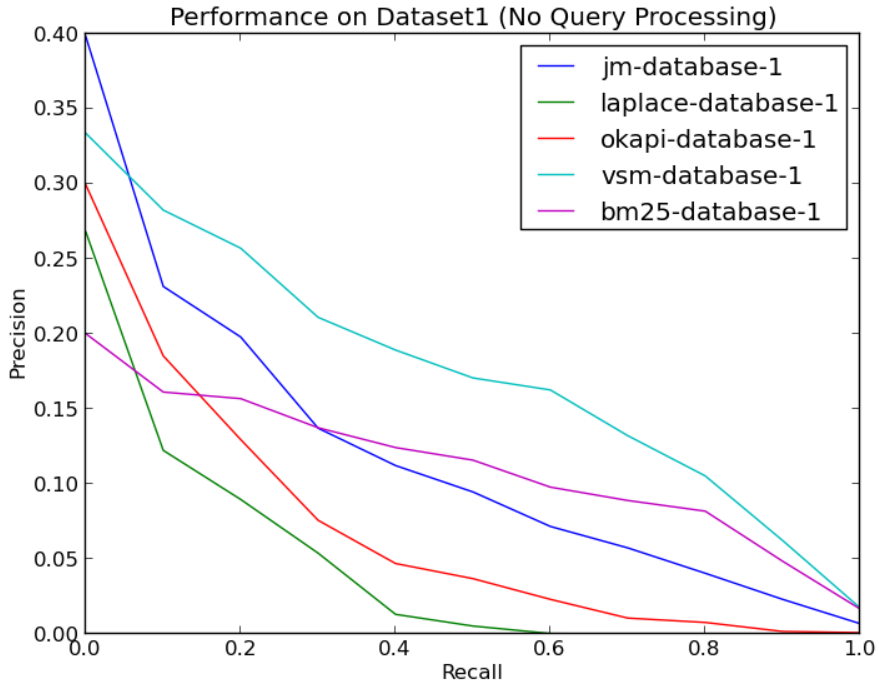

Figure 1.3. Results on Database 1 (No query processing has been done)

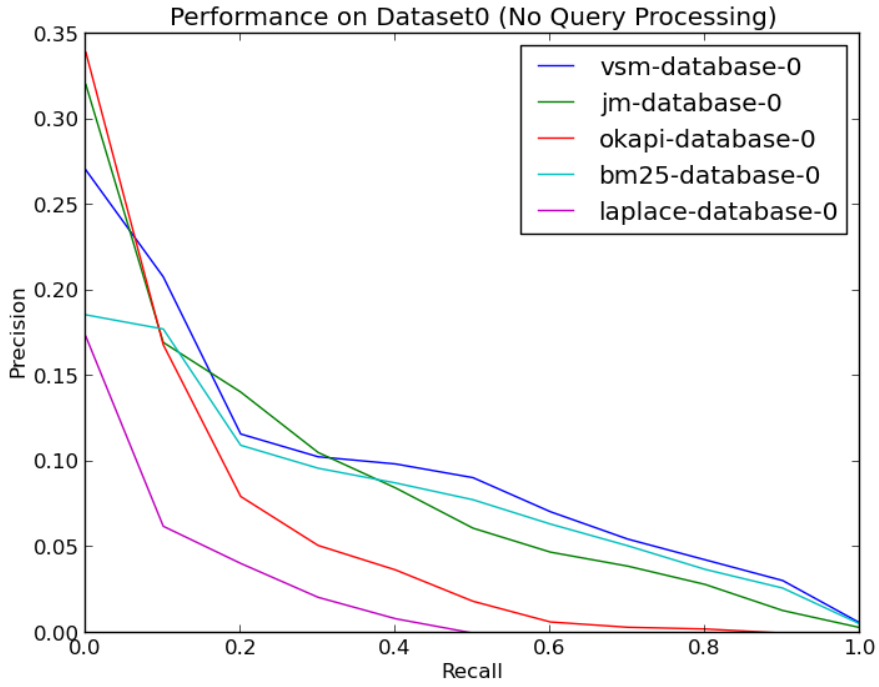

Figure 1.4. Results on Database 0 (No query processing has been done)

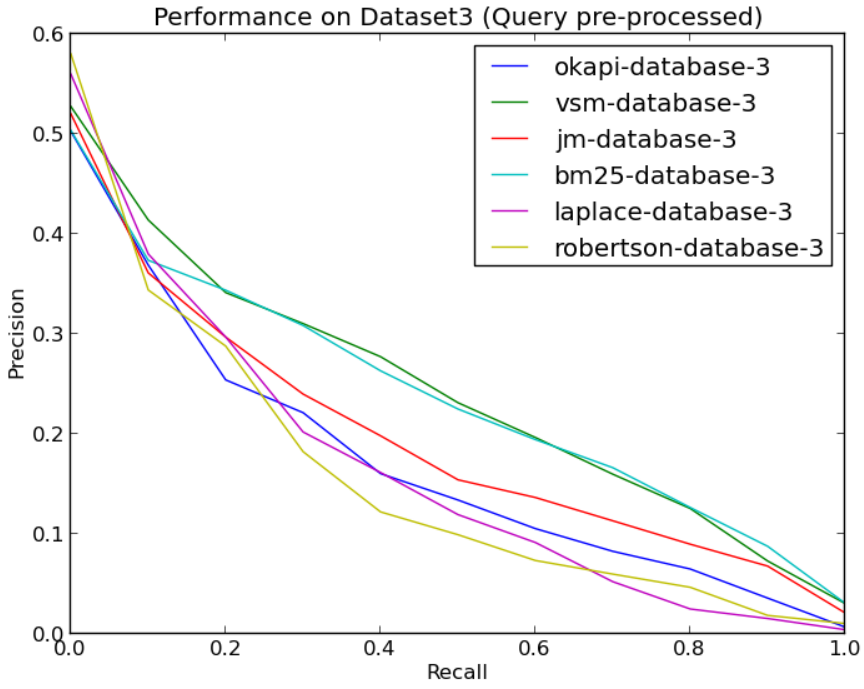

Figure 1.5. Results on Database 3 (Query processing has been done)

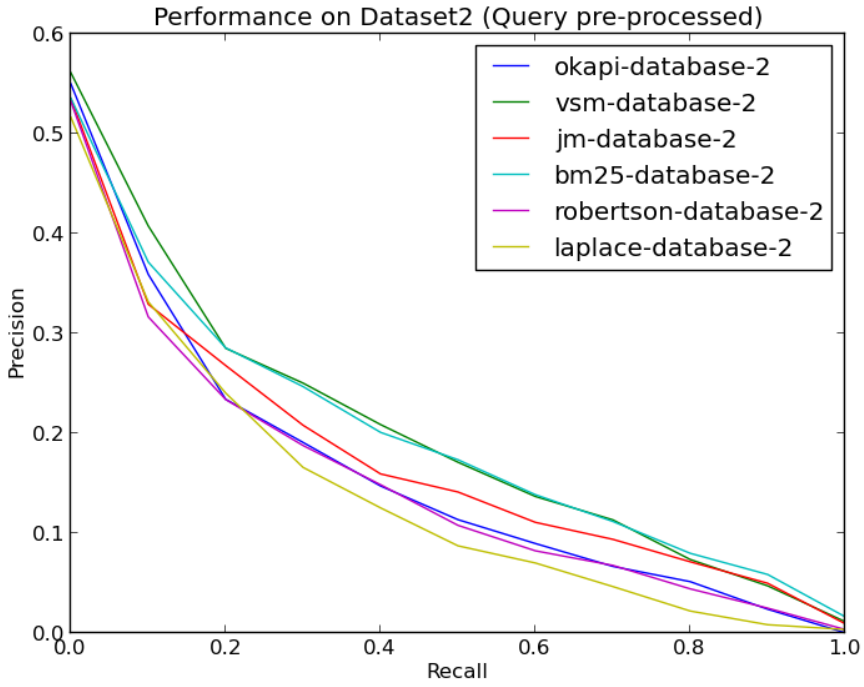

Figure 1.6. Results on Database 2 (Query processing has been done)

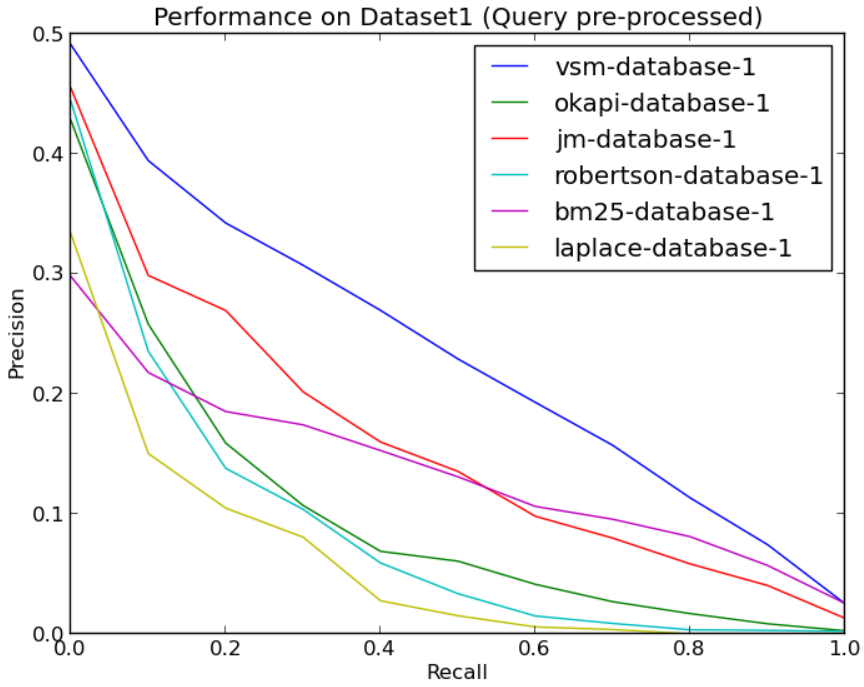

Figure 1.7. Results on Database 1 (Query processing has been done)

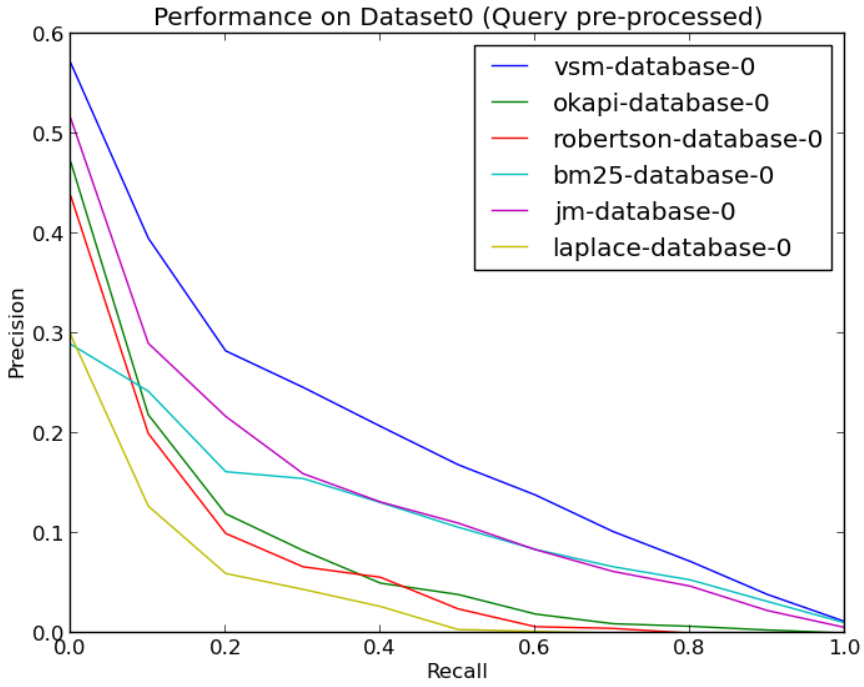

Figure 1.8. Results on Database 0 (Query processing has been done)