Please read this file before running/using the code. The program flow is as below:

→ Create a wordFiles named directory in the same directory you are planning to work at.
→ Start with parse.py
→ Run each of the five models Okapitf, tfidf, Laplace smoothing, Jelenik Mercer, Witten Bell smoothing.

Used Glassgow for cacm collection, its stopwords, qrel file from Search Engine book:
Indexing:

Given corpus has 3204 documents. The documents has the below mentioned sections in them. Not all of the documents have all of the below but definitely all documents have any or all of these:

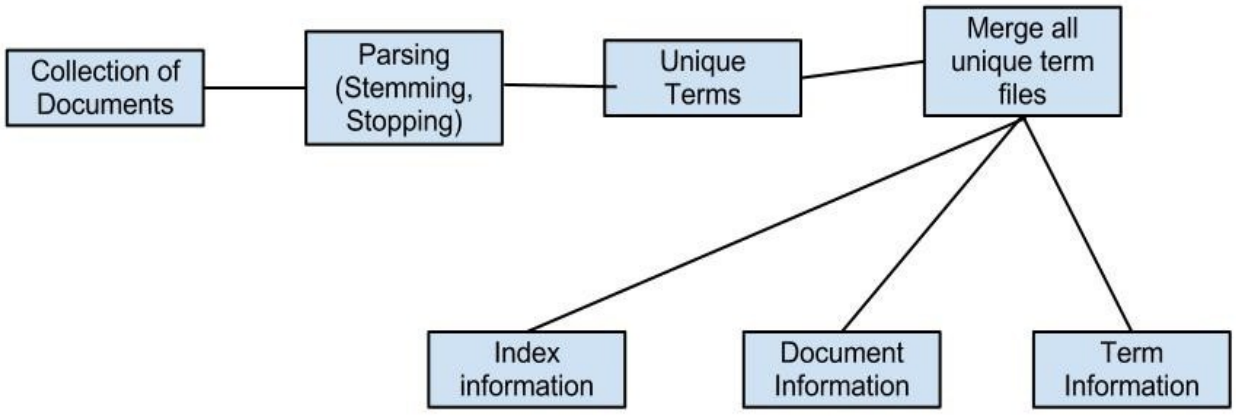
.I (Document #)
.T (Document Title)
.A(Document Author(s))
.W (Document Info)
.B (Bibliography)
.N (Document details)
.C (values)
.K (values)
.X (values → ignored in the project)

On a higher level, the view of the process of the whole project is as below:

-> Parse the documents stop, stem and get all unique words in them and write them as files which contain the document ids in which the file name (term) exists along with its frequency. This is mainly because the program stack memory might not be able to handle all the documents and the corresponding info in real practice.

-> Merge all of these files and make a index file with offset and length for each term. The index file consists of three major files. Each term is assigned a unique id and is maintained through out. The file1 (term information) consists of term id , associated term name, index offset, length and corpus frequency. While file2 (term and document information) consists of term id, corresponding doc id and term freq of the term in the doc and file3 (Document information) consists of document id and its length. Apart from these three main index files, another file which consists of corpus information such as total # of terms, total number of unique terms, total number of documents, document length etc.

-> Once the above files are made, project1's retrieval models are modified in such a way to accept these files as their data structures with corresponding changes and run them to get the values of average



precision and precision at # of documents. Diagramatical representation of the above process is like:

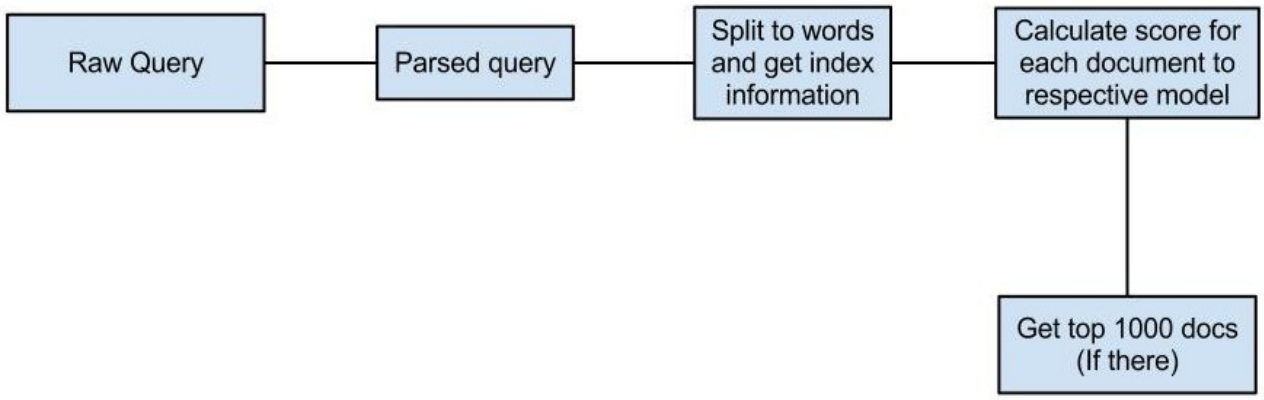
In each of the index files, the sample content that it contains looks like below:

file1 : Term information
termid, its name, index offset, length , corpus freq
ex : 1 preliminari 0 20 20
file2: Term and document information
termid, documentid, termfreq
1 1 1
1 254 1
…
file3: Document information
document id, document length
Observations while creating the above index files:

-> Once the documents are parsed, each of the terms in each document is stopped to eliminate stopped words and then stemmed to get corresponding stem word. By doing this, the words are stored more appropriate for different versions of same word and they are stemmed to the same. Since the given query will also be stemmed irrespective of the given words, this helps to retrieve the relative documents. This is more generic than specific retrieval.
-> The way of parsing the document effects in total number of unique words.
-> Always better to write the terms as files as program memory might not handle or sometimes data might get orderless which is prime in indexing.

Below are the models and their retrieval performance along with description and comparison with proj1 values. For every model, showed the values of the Project 1 and then the current values. Although the corpus is different, the queries are different, we can observe the variations in the values at certain key points such as constant values etc.

On a higher level, the below is the representation of how in each of these models, the score and retrieval is done:



For each of the models, first the values from Project 1 are shown followed by the values from current to identify and analyse the differences. Please note. At the end of all retrieval models, a summary of all

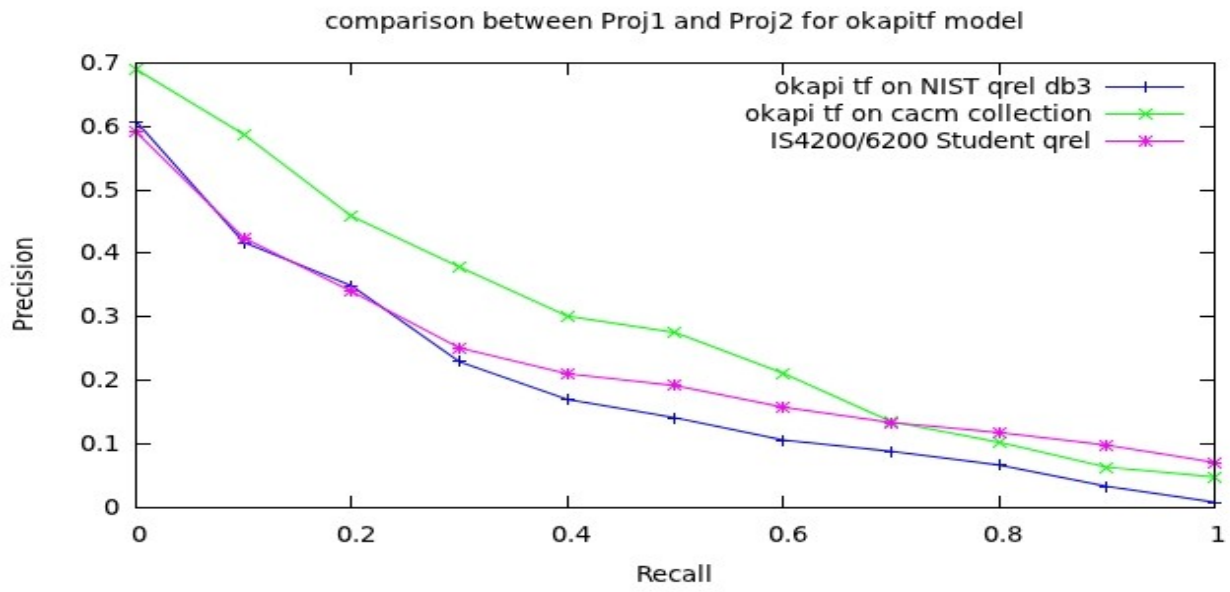
models collective along with their average precision(un interpolated) is given.

-> Okapitf:
The score value is the dot product between document vector and query vector. The querytf is calculated using the formula: tf/(tf+0.5+(1.5*queryLength / avgQueryLength))
while the document tf is calculated using : tf/(tf+0.5+(1.5*docLength/ avgDocLength))
where tf is the term frequency.
This model has the below performance on the lemur collection (From project 1).
Precision values for vector space model for tf only on qrel files:

Qrels	Average Precision
NIST	0.2150
IS4200/6200	0.2396

This model performed decently using the index files on cacm collection: The score for all the 64 queries combined is as below:
Average precision : 0.2775

Average Precision	0.2775
Precision at 10 documents	0.3346
Precision at 30 documents	0.1929
Precision at 60 documents	0.2112
Precision at 90 documents	0.0632



As seen in the graph, okapitf performed well on cacm collection. The reason could be anything from

stopping ,stemming to querying. Though the values are different, they are close.

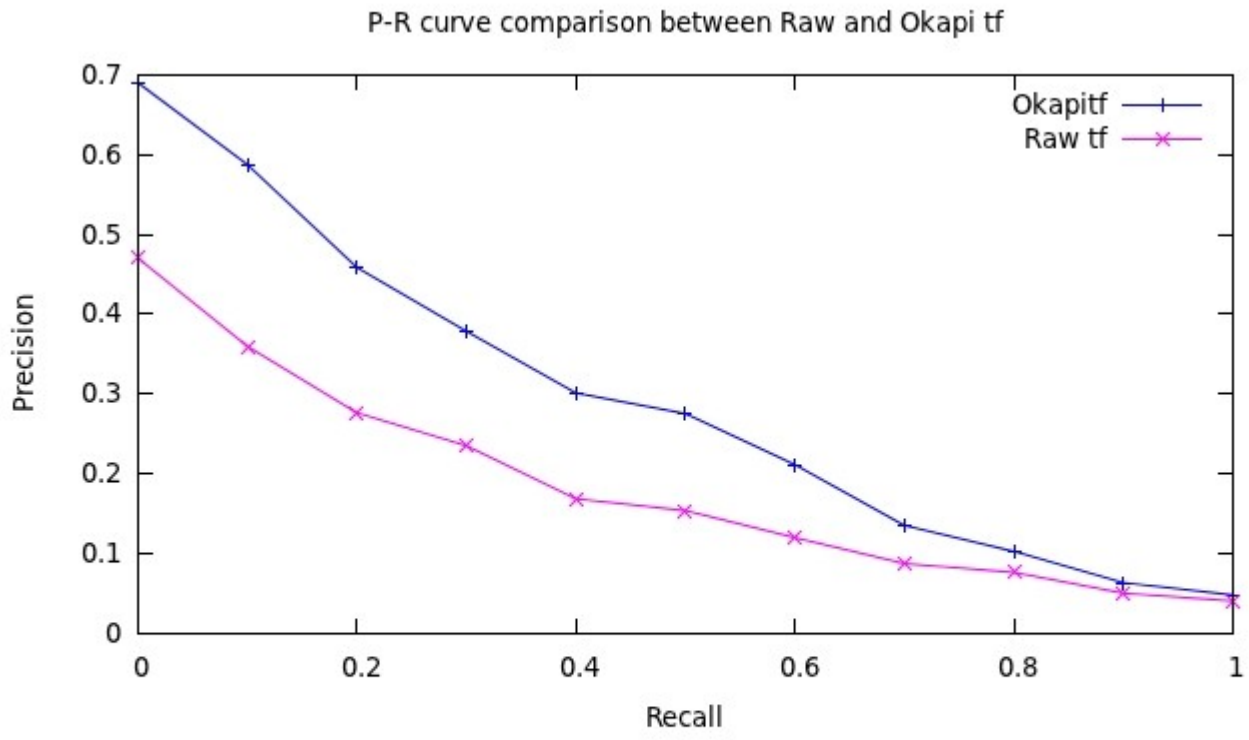
Comparison of recall-precision values of rawtf and okapi tf : **(Extra- credit)**

Raw tf values are as below:

Average Precision	0.1680
Precision at 10 documents	0.3596
Precision at 30 documents	0.2349
Precision at 60 documents	0.1197
Precision at 90 documents	0.0501

The values of rawtf are less than okapi tf as observed. This is because, the score calculation in okapi tf involves a dot product between the querytf and document tf while in the simple rawtf values, the score is just the sum of the query terms in each document.

A graph comparing the values of P-R for rawtf and okapi tf is as below:



Vector space model tf * idf:

Score is calculated in the same way as in tf only except that a factor idf (log N/Nt) is multiplied to the tf. idf indicates the discriminatory power of the term in the space (inverse document frequency). This value is used to compute the weight for each term based on their term frequency in the corpus. This value can be computed for each term since the df value is known upon retrieval.
Precision values for vector space model for tf *idf on qrel files:

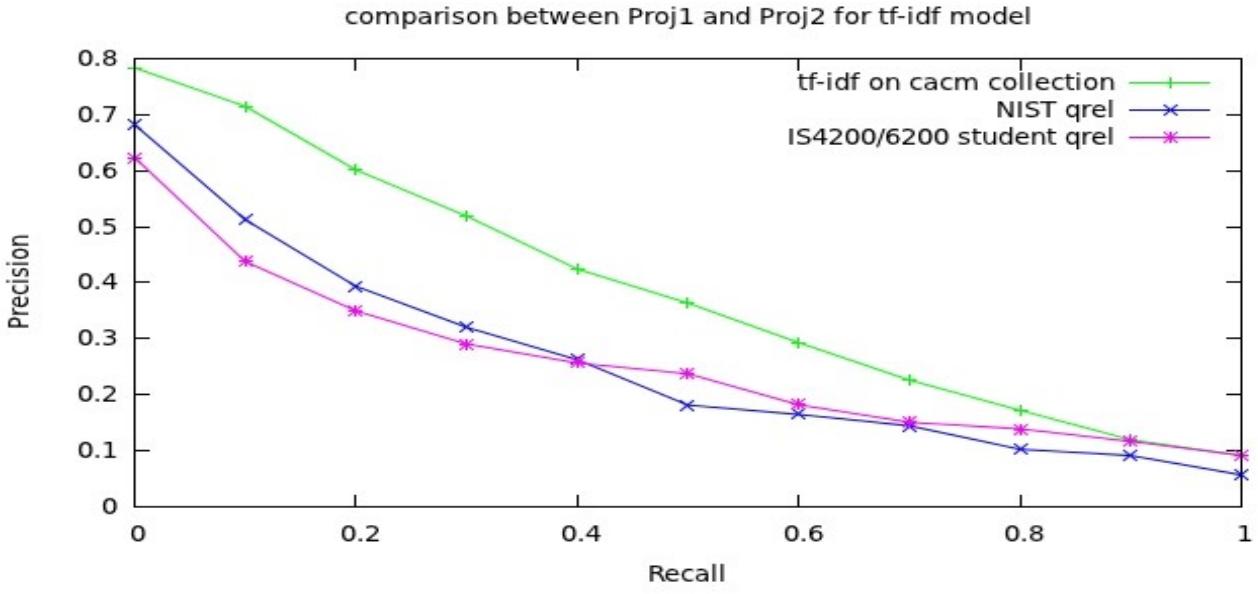
Qrels	Average Precision
NIST	0.2396
IS4200/6200	0.2408

This model performed very well using the index files: The score for all the 64 queries combined is as below:

Average precision : 0.3730

Average Precision	0.3730
Precision at 10 documents	0.7145
Precision at 30 documents	0.5184
Precision at 60 documents	0.2926
Precision at 90 documents	0.1192

The P-R curve comparison graph between Proj1 and current values:



BM25:

BM25 is an effective retrieval model and popular. It adds document and query term weights. This ranking algorithm ranks matching documents according to their relevance to a particular query. Score is calculated using the formula

$$\log \frac{(ri+0.5) / (R-ri+0.5)}{(ni-ri+0.5) / (N-ni-R+ri+0.5)} * \frac{(K1+1) * termfreq}{(K+termfreq)} * \frac{((K2+1) * qtf)}{(K2+qtf)}$$

Score is calculated using the above. The values of K1 and K2 are constant and are considered as K1=1.2 and K2=100 also Ri and ri are considered as 0
Values from Proj1 :

Precision values for vector space model for BM25 on qrel file:

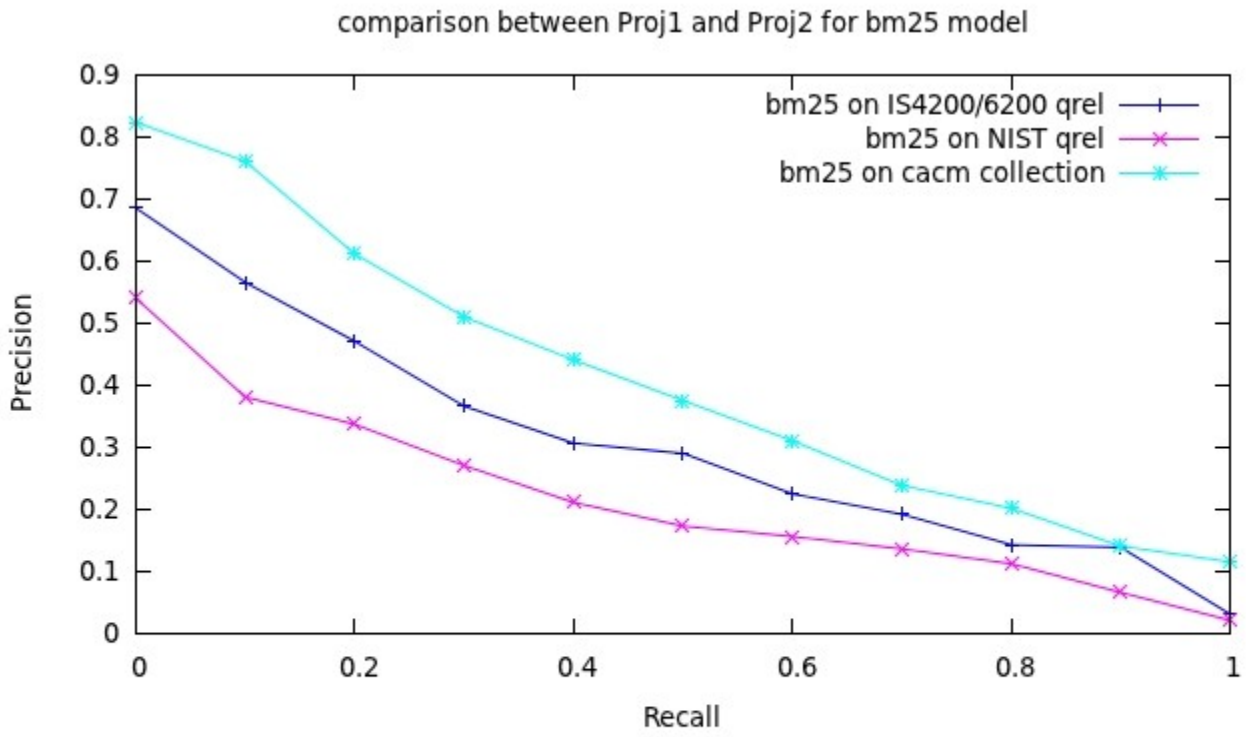
Qrels	Average Precision
NIST	0.2010
IS4200/6200	0.2405

This model performed very well using the index files: The score for all the 64 queries combined is as below: Value of b in this case is 0.75

Average precision : 0.3910

Average Precision	0.3910
Precision at 10 documents	0.7607
Precision at 30 documents	0.5099
Precision at 60 documents	0.3102
Precision at 90 documents	0.1404

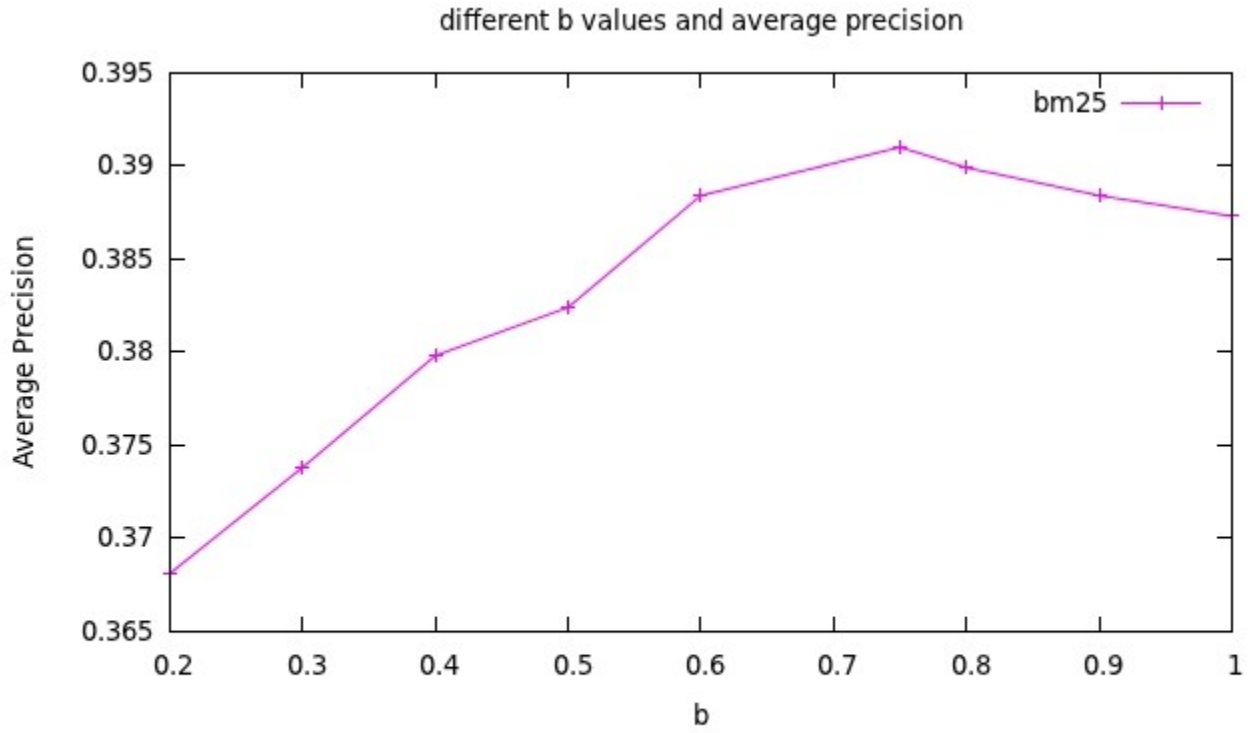
The P-R curve along with comparison graph with Project1 and current values is as below:

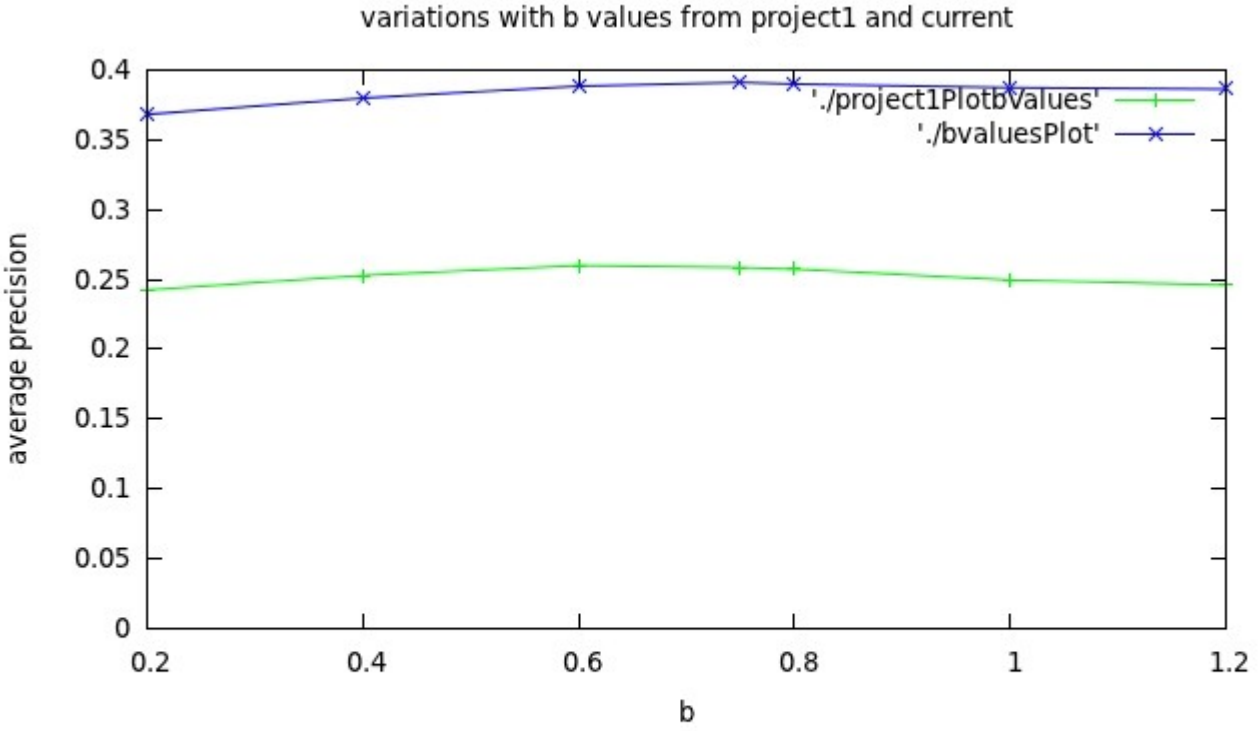


For different values of b, the values of average precision, precision at 10 and 30 docs are:

Value of b	Average precision	Precision at 10 docs	Precision at 30 docs	Precision at 60 docs
0.2	0.3681	0.7294	0.4751	0.2941
0.3	0.3738	0.7352	0.4879	0.2970
0.4	0.3798	0.7441	0.4972	0.3008
0.5	0.3824	0.7459	0.4967	0.3022
0.6	0.3884	0.7535	0.5052	0.3085
0.75	0.3910	0.7607	0.5099	0.3102
0.8	0.3899	0.7621	0.5116	0.3096
0.9	0.3884	0.7566	0.5162	0.3090
1.0	0.3873	0.7519	0.5139	0.3113

Graph between different values of b and corresponding Average precision is as below. The behavior is almost similar to that of project 1 even though the values are different.





The behavior is same between both the projects even though the values are different. At lower values of b, the value is higher, and then keep changing with b values and then finally settle down with minor differences between change in b value.

Laplace smoothing:

Score is calculated with $\sum_{i=\text{no of terms}} pi = mi + 1 / n + k$ where mi is freq count for term and
n is no of terms in document, k is number of unique terms in document.

Values from Proj1 :

Precision values for laplace model on different qrels:

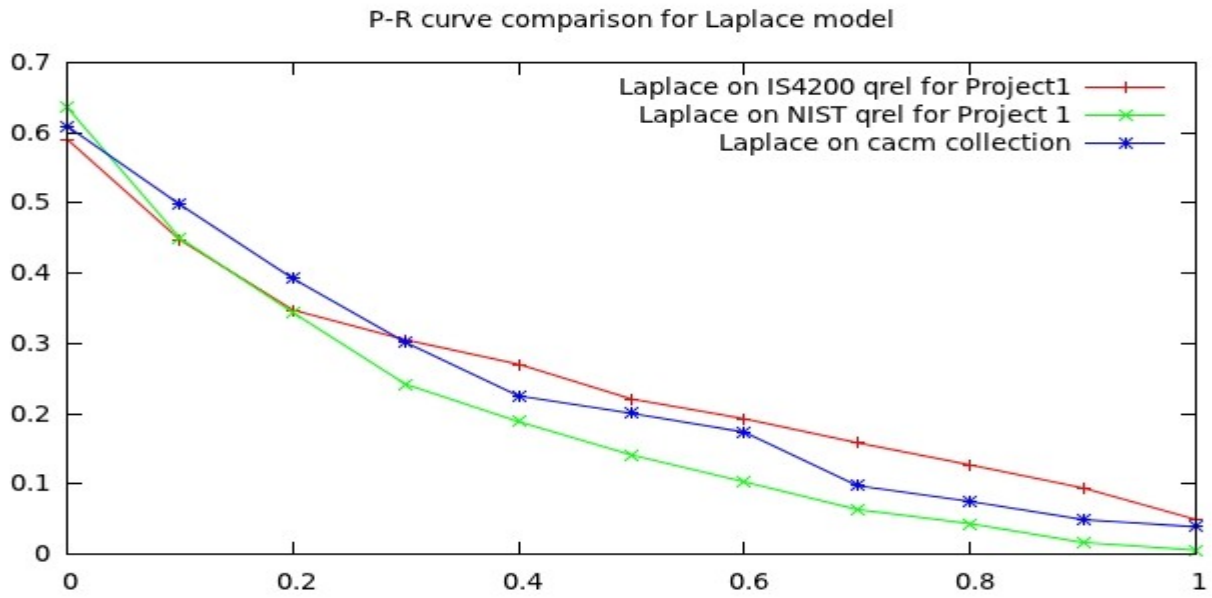
Qrel	Average Precision
NIST	0.1895
IS4200/6200	0.2379

This model performed very low using the index files when compared to the rest models: The score for all the 64 queries combined is as below:

Average precision : 0.2239

Average Precision	0.2239
Precision at 10 documents	0.4976
Precision at 30 documents	0.3013
Precision at 60 documents	0.1737
Precision at 90 documents	0.0493

Laplace performs equally well with that of the project 1 collection. Since Laplace uses probabilistic values for calculating scores, the change in the corpus and the queries doesn't effect much on the value. A graph plotting the P-R curve in comparison of same is in the next page:



Jelenik Mercer:

In this model score is calculated in the same way as in Laplace smoothing except that the probability estimation change in the below way
$\sum$ pi where pi = $\lambda P + (1-\lambda)Q$
i=no of terms

P is estimated probability from document; Q is background probability.

Values from Proj1 :

Average Precision values for vector space model for tf *idf

Qrel	Average Precision
NIST	0.2060
IS4200/6200	0.2395

This model performed decent using the index files: The score for all the 64 queries combined is as below:

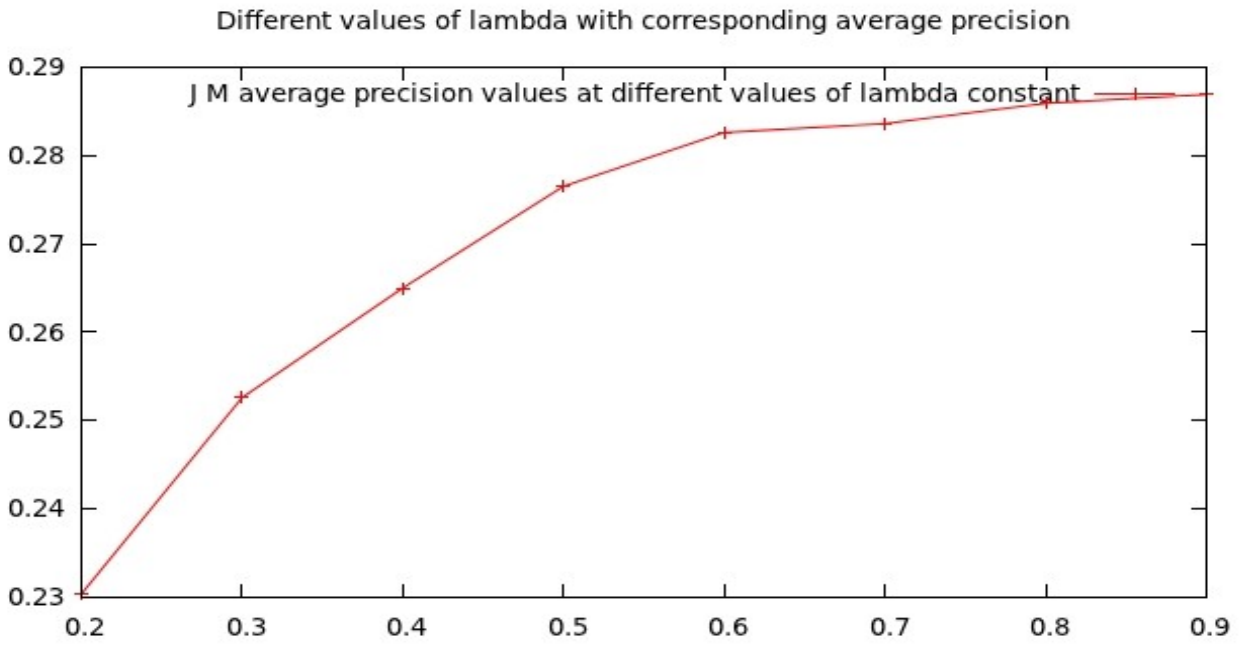
Average Precision	0.2826

Precision at 10 documents	0.5972
Precision at 30 documents	0.3862
Precision at 60 documents	0.2200
Precision at 90 documents	0.0811

For different values of λ, the values of average precision, precision at 10, 30 and 60 docs are:

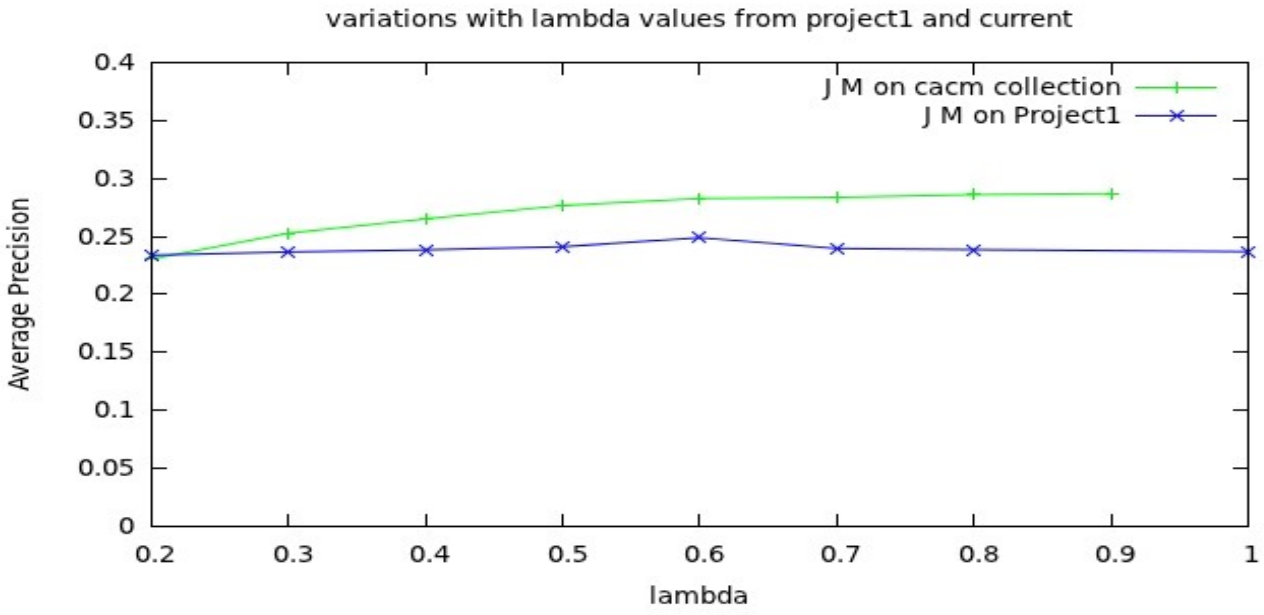
Value of λ	Average precision	Precision at 10 docs	Precision at 30 docs	Precision at 60 docs
0.2	0.2304	0.5007	0.3217	0.1798
0.3	0.2526	0.5410	0.3467	0.1935
0.4	0.2650	0.5622	0.3655	0.2046
0.5	0.2765	0.5950	0.3801	0.2113
0.6	0.2826	0.5972	0.3862	0.2200
0.7	0.2836	0.5800	0.3831	0.2314
0.8	0.2859	0.5792	0.3936	0.2349
0.9	0.2869	0.5875	0.3939	0.2348

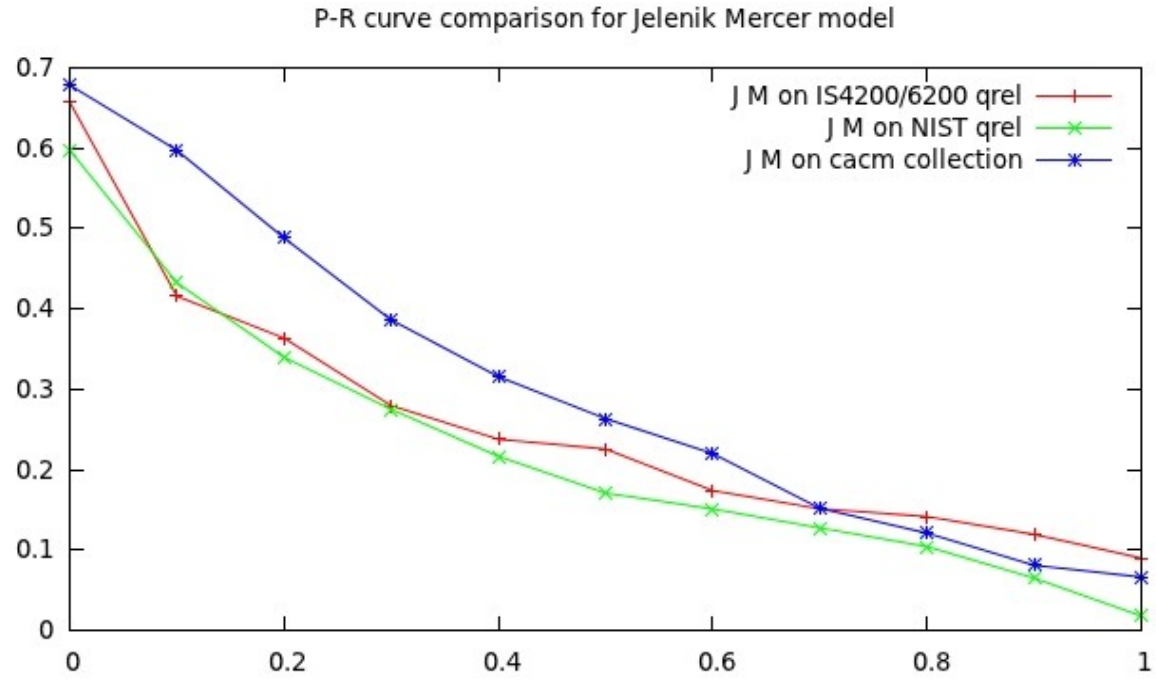
The P-R curve is as below at constant 0.8 along with comparison with Project1 values:



At different values of lambda , its average precision (uninterpolated) is plotted as below:

At different values of lambda, the average precision for proj1 also behave differently. Comparing both the values doesn't show a big difference between current project but there is. Below is the curve with lambda values and average precision values for both the project J Mercer retrieval performances.

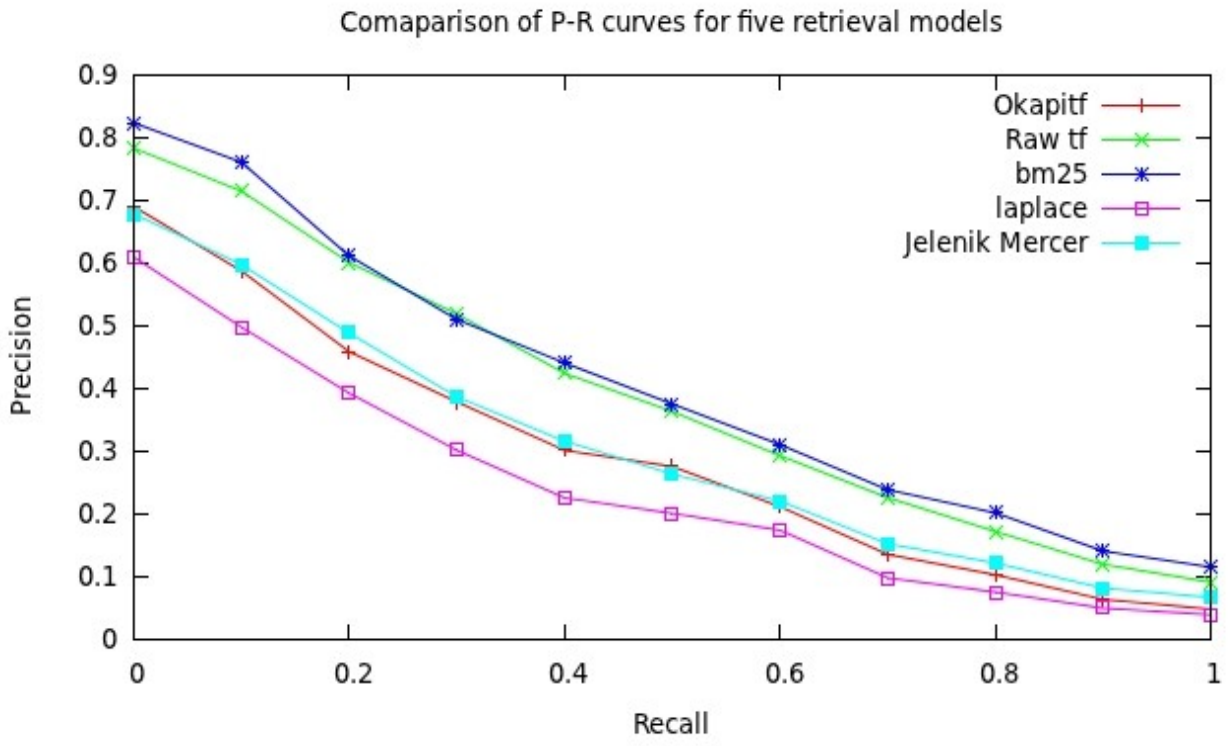




Summary of all average precision values and precision at 10 and 30 docs for all retrieval models:

Model	Average Precision (Uninterpolated)	Precision at 10 docs (Interpolated)	Precision at 30 docs (Interpolated)
Okapi tf	**0.2775**	**0.5873**	**0.3783**
Tf * idf	**0.3730**	**0.7145**	**0.5184**
BM 25	**0.3910**	**0.7607**	**0.5099**
Laplace Smoothing	**0.2239**	**0.4976**	**0.3013**
Jelenik Mercer	**0.2859**	**0.5792**	**0.3936**
Witten Bell (Below)	**0.3152**	**0.6329**	**0.4247**

Comparison graph of P-R curves for all five retrieval models for cacm collection:



Extra – credit:

→ As seen above computed Raw tf and plotted P-R graph in comparison with Okapi tf. The values are analysed and plotted above in the okapi tf section.

→ Cosine similarity.

Score value is calculated using the document and query magnitude.

Similarity = Q. D / ||Q|| . ||D||

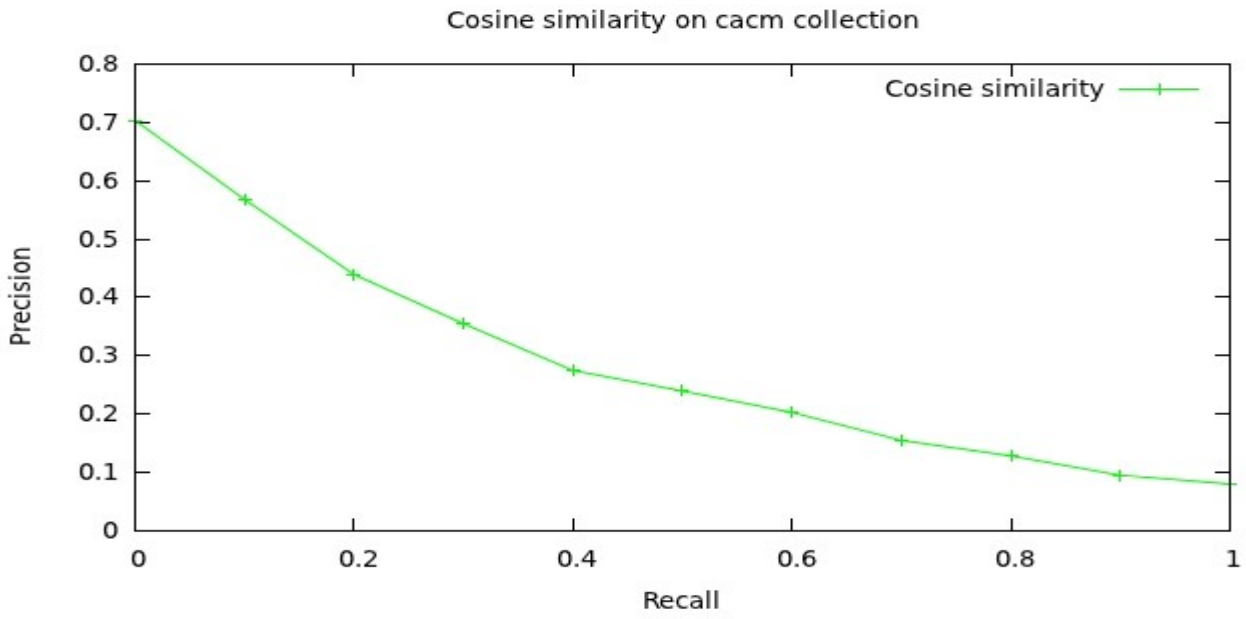
This calculation was not possible in the project1 because there is no information about the # of unique terms in each document. Now since we have the information, I am able to calculate the score.

Below are the values of the average precision and precision at 10,30 and 60 docs.

Average precision : 0.2747

Average Precision	0.2747
Precision at 10 documents	0.5677
Precision at 30 documents	0.3546
Precision at 60 documents	0.2024
Precision at 90 documents	0.0946

Graph of P-R curve for cosine similarity is as below:



→ Witten Bell smoothing:

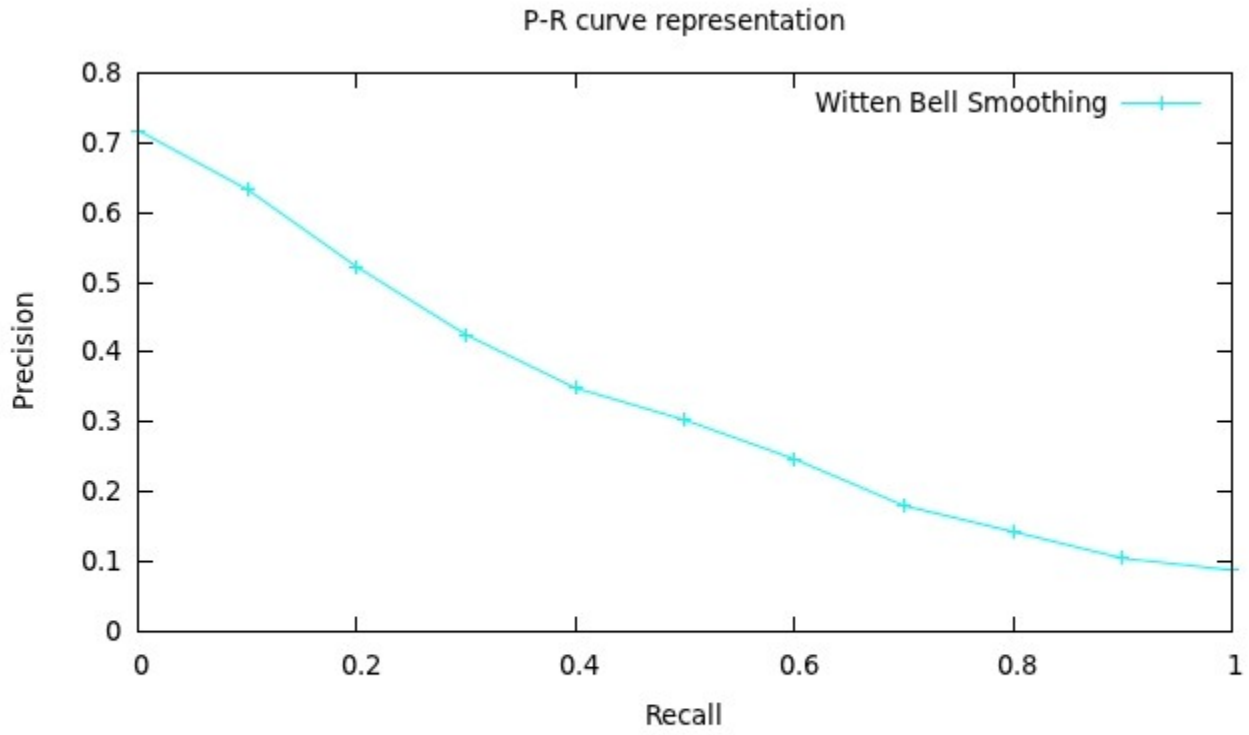
This smoothing technique calculates the score value as any other language model but the constant used is calculated using the document length and number of unique terms in that document. This brings up the difference between this technique and others.

This model performed well using the index files: The score for all the 64 queries combined is as below:

Average precision : 0.3152

Average Precision	0.3152
Precision at 10 documents	0.6329
Precision at 30 documents	0.4247
Precision at 60 documents	0.2467
Precision at 90 documents	0.1044

The P-R curve is as below:



Dirichlet Smoothing:

This smoothing technique calculates the score based on a constant assumed. The value of constant depends on the corpus size. Below are the values of average precision and precision at 10 and 30 documents at u=2000:

Average precision : 0.3078

Average Precision	0.3078
Precision at 10 documents	0.6952
Precision at 30 documents	0.5087
Precision at 60 documents	0.2567
Precision at 90 documents	0.0933

At different values of u, the variation in uninterpolated average precision:

Value of u	Average Precision
500	0.3296
1000	0.3196
1500	0.3106
2000	0.3078
2500	0.3025

→ Indexing on Author, Title:

A separate index on author names from each file is created. This helps to search all documents given the author name. Creation of index files is same as with above but each of the sectionsa re seperately parsed to get the corresponding information. Please see code sent for the same. With the way of the code written, indexing can be done on any part of the document. The index files are created and queried from any of the retrieval models the same way the main indexing is done.

→ Analysis of retrieval techniques and query processing:

→ BM25 has highest average precision. The score value depends on the constants we choose. The description above has shown precision values at different constant b values. The BM25 values look more consistent on both the files than any other retrieval model. This ranking algorithm is more efficient than others.

→ Queries which are parsed has more relevance in retrieved documents than queries which are not stopped and stemmed.

→ While the okapi methods are term independent, the language models consider the whole query which makes more sense. The language models are more relevant but the calculation is tedious as it involves very small and negative values including logs.

→ The unique terms depend on stopping and stemming. If zipf's law is used to remove the top most terms which appear in many other docs, then performance of the retrieval system raises by a point