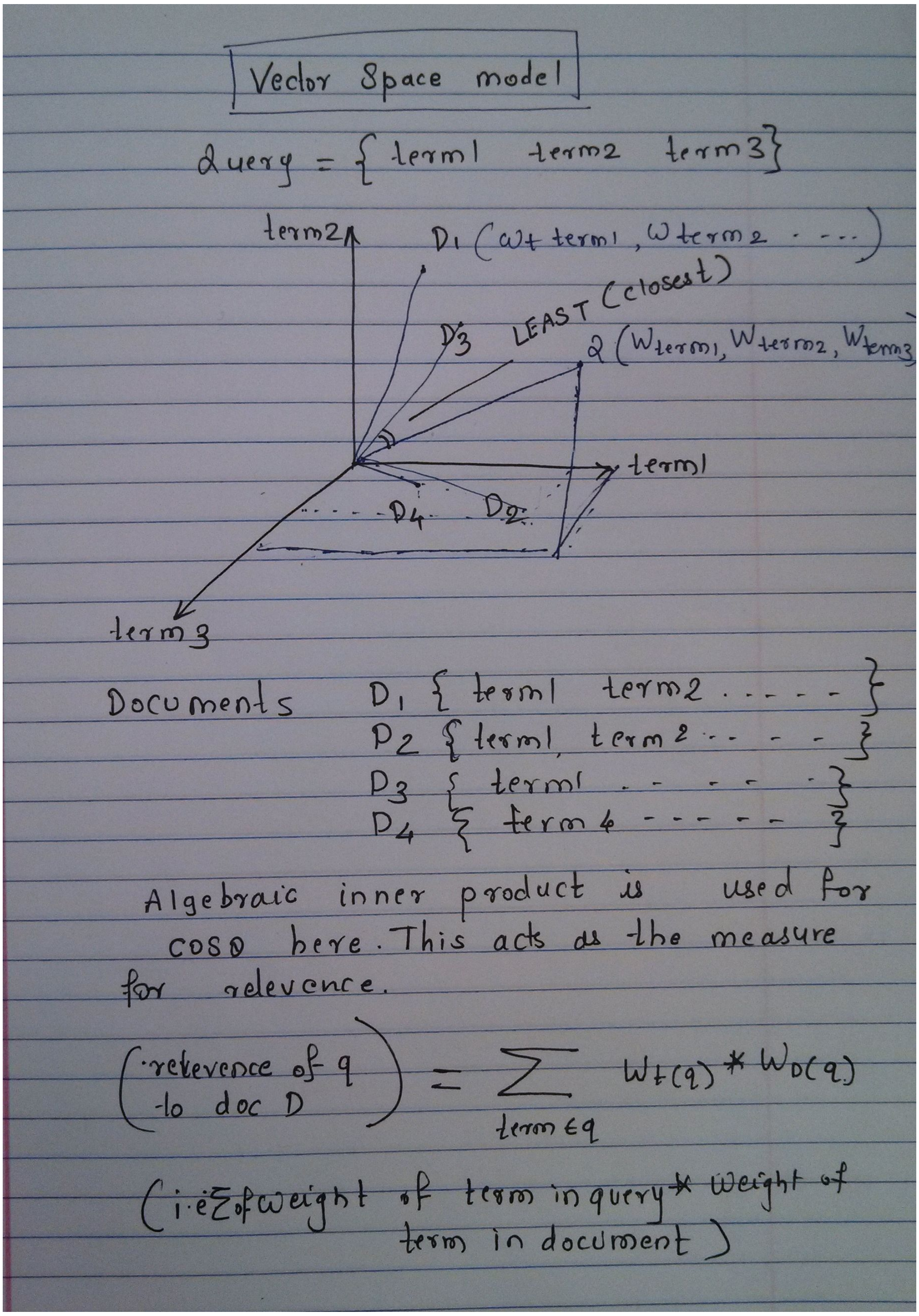
Building an IR system: Report – by Poonam Bhide

Various models have been proposed for retrieving the information from unstructured document. For this project implemented the following retrieval models on the Lemur database. This analysis report first gives theoretical details of every retrieval model and factors that can affect performance of retrieval. Later, results actually obtained and analysis is performed.

MODEL 1 : Vector Space Model



This model captures the relative importance of the term in the document. The terms are represented along the axes. The document and queries are represented as vectors and the relevance is based on the cosine similarity between those vectors. The vector space model behaves in the follows:
Assuming query is of 3 terms (as it is possible to visualize the vectors in 3D). From the figure above, D3 is closest to Q as the angle between them is least. Hence it is most relevant.

In query Q and Document, some weight is assigned to every term. It can be done in multiple ways.

1. Weight of term in (query /document) = Term Frequency (query / document)

In this case every term is equally important in the document and documents with more length are likely to be more relevant hence it is not considered to be effective.
In this project implement OKTF vector space model was implemented.
OKTF (Robertson's TF) was used for the term weights in query and document.

OKTF = tf/(tf + 0.5 + 1.5*doclen/avgdoclen)

In this case the length of document is normalized due to factor (doclen/avgdoclen) hence the relevance of the query will not depend upon length of documents.
Also taking the algebraic inner product approximated cosine similarity.
Following results were obtained after running trec script on one of the qrel files :

Queryid (Num): 25
Total number of documents over all queries
Retrieved: 25000
Relevant: 560
Rel_ret: 440
Interpolated Recall - Precision Averages:
at 0.00 0.5552
at 0.10 0.3696
at 0.20 0.2988
at 0.30 0.2241
at 0.40 0.1876
at 0.50 0.1647
at 0.60 0.1537
at 0.70 0.1343
at 0.80 0.1207
at 0.90 0.0963
at 1.00 0.0662

Average precision (non-interpolated) for all rel docs(averaged over queries)
0.1961
Precision:
At 5 docs: 0.2640
At 10 docs: 0.2080
At 15 docs: 0.1867
At 20 docs: 0.1760
At 30 docs: 0.1573
At 100 docs: 0.0948
At 200 docs: 0.0606
At 500 docs: 0.0314
At 1000 docs: 0.0176
R-Precision (precision after R (= num_rel for a query) docs retrieved):
Exact: 0.2065

From the results obtained,

Recall = No. of relevant documents retrieved / No. of actual relevant documents
= 440/560
= 0.78571428571 = 78.5%
Precision =0.2065

From the precision value calculated after no. of documents is decreasing as more and more terms are encountered , no. Of irrelevant document increases, hence the precision goes on decreasing.
This overall approach of vector space model is fetching pretty decent results but it has some drawbacks:

1. The order in which the terms appear in the document is lost in the vector space representation as this is bag of words model.
2. Theoretically assumes terms are statistically independent.
3. Weighting is intuitive but not very formal.
4. Also it does not consider higher weight for terms that occur in few documents hence does not assign higher weight to more specific word.

To overcome the 4th drawback mentioned above Inverse Document Frequency (IDF) was multiplied with OKTFs. Details are discussed in model 2 below.

MODEL 2: IDF

Effects of introducing IDF :

IDF = log (N/DF)

Where N = no. of documents in the collection
DF =document frequency of term in the collection

Due to IDF, terms in query that are in few documents get multiplied by a higher IDF hence overall weight of document containing that term increases.

Example : If Query is Linux Kernel,

Chances of getting Linux in documents are more than Kernel as Linux is more general term compared to Kernel. Hence Document frequency of Linux would be more say 1000.Document frequency of kernel is 50. If N=100000

IDFLINUX =log(100000/1000)=2

IDFKERNEL=log (100000/50) = 3.3 (Higher weight)

Hence it would increase the weight of Kernel and chances of getting documents containing kernel will be higher.[4]

The formula gets slightly modified:
Weight is = OKTF *IDF
For the code IDF is considered to be equal to **IDF = log (N/(1+DF)** as DF=0 can lead to illegal value. Following are the results obtained through Trec Evaluation (Student version):

Queryid (Num): 25
Total number of documents over all queries
Retrieved: 25000
Relevant: 560
Rel_ret: 530
Interpolated Recall - Precision Averages:
at 0.00 0.6296
at 0.10 0.4917
at 0.20 0.4214
at 0.30 0.3331
at 0.40 0.3081
at 0.50 0.2874
at 0.60 0.2422
at 0.70 0.2075
at 0.80 0.1901
at 0.90 0.1438
at 1.00 0.1104
Average precision (non-interpolated) for all rel docs(averaged over queries)
0.2832
Precision:
At 5 docs: 0.3120
At 10 docs: 0.2920
At 15 docs: 0.2613
At 20 docs: 0.2540
At 30 docs: 0.2267
At 100 docs: 0.1300
At 200 docs: 0.0818
At 500 docs: 0.0392
At 1000 docs: 0.0212
R-Precision (precision after R (= num_rel for a query) docs retrieved):

Exact: 0.2887

Recall = 530/560= 0.94 = 94%

Precision : : 0.2887

As compared to model 1 ,the precision and recall both the values are improved because of IDF.

Models 1 and 2 were typical Information Retrieval models where given a query we find it's relevance with the document based on some methodology. Language Modeling is based on the idea: "A common suggestion to users for coming up with good queries is to think of words that would likely appear in a relevant document, and to use those words as the query." In the language modeling approach to IR directly models that idea: a document is a good match to a query if the document model is likely to generate the query, which will in turn happen if the document contains the query words often.[2]

MODEL 3: The maximum query likelihood model and Laplace smoothing

It is a probabilistic model, in which random process is generation of queries. Maximum query likelihood is, given a document what are the chances it will generate the query. Hence the aim is to rank documents by P(d|q) i.e. probability of generating q.

As per Naïve Bayes Theorem,

$$P(d \mid q) = P(q \mid d)P(d)/P(q)$$

Prior P(d) is ignored and P(q) is uniform. Assuming terms are independent.
Hence P(d|q) is calculated based upon probability of query q under document d.
Applying Multinomial unigram language model M_d gives:

$$P(q \mid Md) = Kq \prod P(t \mid Md)tft,d$$

Where Kq = Ld!/(tft1,d!tft2,d! · · · tftM,d!) is the multinomial coefficient for the query q. As it is constant for q it can be ignored.

Hence Maximum Likelihood (MLE) under unigram assumption becomes :

$$P(q \mid M_d) = \prod_{(t \,\varepsilon\, q)} Pmle(t \mid M_d)$$

$$= \prod_{(t \,\varepsilon\, q)} (tf_{t,d} \;/\; L_d)$$

i.e. Term Frequency of each term in the query is divided by length of document L_d. The product of these probabilities will be 0 if the numerator i.e. term frequency of any term is 0 in the document. Hence for that some smoothing techniques have been proposed. In first approach in model 3, Laplace smoothing was used. Laplace smoothing is an additive smoothing in which is of the form as below:

$$\hat{P}(t|d) = \frac{tf_{t,d} + \alpha \hat{P}(t|M_c)}{L_d + \alpha}$$

As per Laplace smoothing, for the model 3 following formula was used:

$p_i = (c_i + 1) / (n + k)$

Where ci=freq count (tf) of i, n=number of terms in document (doc length) ,
k=number of unique terms in corpus.
Due to this very small probability of a word that occurs in query and does not occur in document gets multiplied. The factor by which it gets multiplied is 1 / (n+k). The purpose of smoothing techniques is not nly to adjust zero probabilities but also give better weights to some terms.

For this model summation of log p_i is taken
Following are the results obtained through Trec Evaluation (Student version):

Queryid (Num): 25
Total number of documents over all queries
Retrieved: 25000
Relevant: 560
Rel_ret: 459
Interpolated Recall - Precision Averages:
at 0.00 0.5756
at 0.10 0.3477
at 0.20 0.3005
at 0.30 0.2482
at 0.40 0.2044
at 0.50 0.1815
at 0.60 0.1587
at 0.70 0.1412
at 0.80 0.1191
at 0.90 0.1027
at 1.00 0.0705
Average precision (non-interpolated) for all rel docs(averaged over queries)
0.2043
Precision:
At 5 docs: 0.3040
At 10 docs: 0.2400
At 15 docs: 0.2293
At 20 docs: 0.2280
At 30 docs: 0.1920
At 100 docs: 0.1052
At 200 docs: 0.0658
At 500 docs: 0.0330
At 1000 docs: 0.0184

R-Precision (precision after R (= num_rel for a query) docs retrieved):
Exact: 0.1968
Total Recall = 459/560

= 0.8196 = 82 % (approx)
Precision : 0.1968

MODEL 4: Jelinek-Mercer smoothing[2]

Other technique for smoothing that was used was based upon giving a non-zero probability to words that are in query and appear in documents is based upon the entire collection. The general approach is that a non-occurring term should be possible in a query, but its probability should be somewhat close to but no more likely than would be expected by chance from the whole collection. That is, if tft,d = 0 then

$\hat{P}(t|Md) \leq cft/T$

where cft is the raw count of the term in the collection, and T is the raw size (number of tokens) of the entire collection. A simple idea that works well in practice is to use a mixture between a document-specific multinomial distribution and a multinomial distribution estimated from the entire collection:

$\hat{P}(t|d) = \lambda \hat{P}mle(t|Md) + (1 - \lambda) \hat{P}mle(t|Mc)$

where $0 < \lambda < 1$ and Mc is a language model built from the entire document collection. [2]

Hence for the terms that are not the document, the collection frequency of the term over entire collection gets added up.The formula for Jelinek-Mercer smoothing is:

$p_i = \lambda P + (1 - \lambda) Q$

where P is the estimated probability from document (max likelihood = ci/n) and Q is the estimated probability from corpus. For simplicity, you may use background probability = cf / total number of terms in the corpus). . The performance of this model would depend on value of λ. For this model was tested on 25 queries for 3 values of lambda. Comparison and analysis is presented in another document. Following results were obtained on trec when tested on student qrel file and λ=0.2.

Queryid (Num): 25
Total number of documents over all queries
Retrieved: 25000
Relevant: 560

Rel_ret: 472
Interpolated Recall - Precision Averages:
at 0.00 0.4613
at 0.10 0.3620
at 0.20 0.3103
at 0.30 0.2475
at 0.40 0.2366
at 0.50 0.2053
at 0.60 0.1654
at 0.70 0.1525
at 0.80 0.1357
at 0.90 0.1156
at 1.00 0.0782
Average precision (non-interpolated) for all rel docs(averaged over queries)
0.2035
Precision:
At 5 docs: 0.2400
At 10 docs: 0.2000
At 15 docs: 0.1840
At 20 docs: 0.1800
At 30 docs: 0.1787
At 100 docs: 0.1064
At 200 docs: 0.0666
At 500 docs: 0.0335
At 1000 docs: 0.0189
R-Precision (precision after R (= num_rel for a query) docs retrieved):
Exact: 0.1919

Recall = 472/560 =0.8428 =84.28%
Precision = 0.1919

Probabilistic model are completely dependent on the probability theory where as vector space model used earlier was based on common experiments in order to find the relevance between queries and documents. Probabilistic model assumes an ideal answer set for each model and by iterative changing certain parameters precision recall are changed to find optimum value. Some assumptions of this maximum likelihood approach are :

- term independence
- terms not in the query don't affect the outcome
- document relevance values are independent

In general a better performance approach for probabilistic model was proposed called BM25.

MODEL 5: BM25 [1]

BM 25 is based upon binary independence model. It is based upon the Baye's rule which states that Document D is relevant if the probability of relevance is greater than of non-relevance. (i.e. P(R|D)>P(NR|D)). Applying Baye's Theorem, we get :

$$\frac{P(D|R)}{P(D|NR)} > \frac{P(NR)}{P(R)}$$

It is calculated as follows:

$$\frac{P(D|R)}{P(D|NR)} = \prod_{i:d_i=1} \frac{p_i}{s_i} \cdot \prod_{i:d_i=0} \frac{1-p_i}{1-s_i}$$

Which is eventually :

$$\sum_{i:d_i=1} \log \frac{p_i(1-s_i)}{s_i(1-p_i)}$$

The final formula used in BM 25 is :

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

– k1, k2 and K are parameters whose values are set empirically

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

where dl = document length

– Typical TREC value for k1 is1.2, k2 varies from 0 to1000,b =0.75

Following were results obtained from BM25:
Queryid (Num): 25
Total number of documents over all queries
  Retrieved: 25000
  Relevant: 560
  Rel_ret: 515
Interpolated Recall - Precision Averages:
  at 0.00 0.5520
  at 0.10 0.4097
  at 0.20 0.3777
  at 0.30 0.3070
  at 0.40 0.2775
  at 0.50 0.2437
  at 0.60 0.1922
  at 0.70 0.1714

at 0.80 0.1503
at 0.90 0.1100
at 1.00 0.0807
Average precision (non-interpolated) for all rel docs(averaged over queries)
0.2426
Precision:
At 5 docs: 0.2880
At 10 docs: 0.2680
At 15 docs: 0.2427
At 20 docs: 0.2400
At 30 docs: 0.2120
At 100 docs: 0.1208
At 200 docs: 0.0764
At 500 docs: 0.0381
At 1000 docs: 0.0206

R-Precision (precision after R (= num_rel for a query) docs retrieved):
Exact: 0.2321
Recall = 515/560
=0.91
=91%
Precision = 0.2321

Precision and Recall Analysis

Precision is the measure of quality and Recall is the measure of quantity.

Precision = No. Of Relevant Retrieved Documents/ Retrieved Documents
i.e. what is the percentage of relevant documents in the retrieved documents. Higher precision means showing least number of irrelevant documents.
In Lemur, if any one term is found in the document D,D gets added into the set of relevant documents. This will lead to addition of many irrelevant documents. From those possible relevant documents, top 1000 ranked documents are taken. Hence the precision value of the overall collection is low.

Recall = No. Of Relevant Retrieved Documents / Actual Relevant Documents
This is a quantity metric, which shows what percentage of relevant documents was actually retrieved.
For Lemur, chances of getting high recall is more because every term in the query is checked and documents containing the term are possible relevant documents. Hence chances of getting higher recall are more.

Ideally the precision and recall both should be higher. In Lemur, due to some constraints put on the implementation it is not showing both the values on higher side.

Methodical Comparison and Recall, Precision based Analysis

Following table shows the brief analysis of every model ,recall and precision value based on the IS4200/CS6200 qrel. Later Analysis for 10,30 documents is provided and trends are discussed. Later the analysis of models based on number of queries is evaluated. All the evaluations were done on d=3.

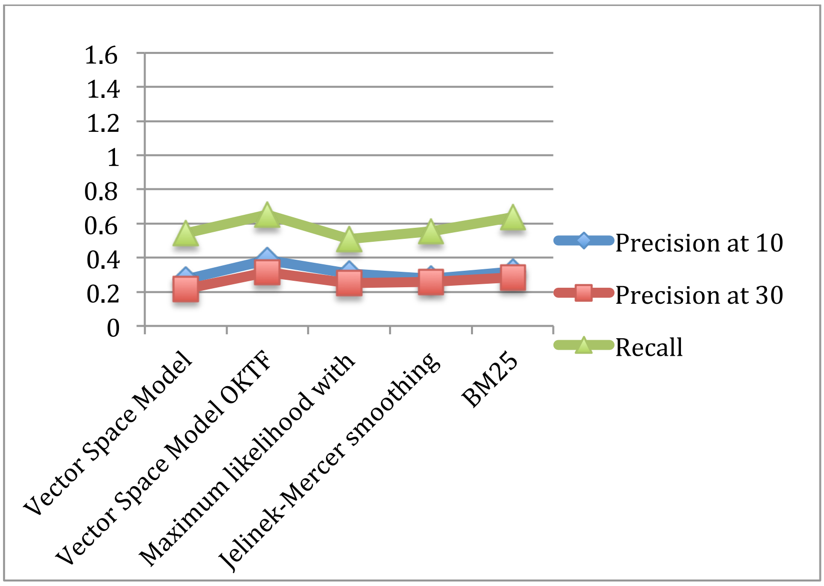
Method wise details (based on 1 qrel) :

Model	Type	Key Feature	Recall	Precision
Vector Space model	Non-Probabilistic (Ranking based on term weights in query and documents	OKTF i.e. Weight given to term in query and document. It depends on doclen/avg/doclen	0.78	0.20
Vector Space model with IDF	Non-Probabilistic (Ranking based on term weights in query and documents and IDF of considered	IDF (Terms with less document frequency give higher weight to document that contains the term)	0.94	0.28
Maximum Likelihood, Laplace Smoothing	Probabilistic model based on probability of generating a query given a document and smoothing is additive smoothing	It is based upon term frequency over document length. The probabilities of term that are present in query and not in document get some less probability.	0.82	0.19
Jelinek-Mercer smoothing	Probabilistic model based on probability of generating a query given a document. Smoothing is based upon the collection model as well.	Mc and λ Collection model and value of λ typically affect the performance.	0.84	0.19
BM25	Probabilistic model based on Binary Independence model where relevance is decided by the higher probability of relevant document over non-relevant document	The performance would vary based on K1,K2, and b	0.91	0.23

Results of Evaluation after 10,30 Documents for NIST qrel :

Model	NIST QREL (Total Documents 1832)		
	Precision at 10	Precision at 30	Recall
Vector Space Model (OKTF)	0.272	0.2173	0.544
Vector Space Model OKTF with IDF	0.384	0.316	0.652
Maximum likelihood with Laplace Smoothing	0.308	0.2493	0.5076
Jelinek-Mercer smoothing	0.276	0.26	0.553
BM25	0.32	0.2853	0.635

Table 1

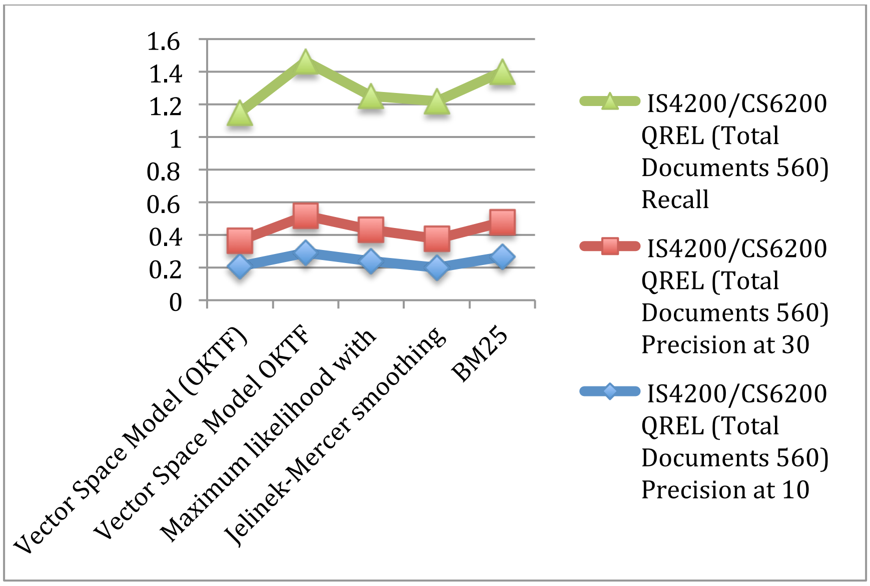


Graph 1: Precision, Recall for all models for NIST

Results of Evaluation after 10,30 Documents for IS4200/CS6200 qrel:

Model	IS4200/CS6200 QREL (Total Documents 560)		
	Precision at 10	Precision at 30	Recall
Vector Space Model (OKTF)	0.208	0.1573	0.7857
Vector Space Model OKTF with IDF	0.292	0.2267	0.9464
Maximum likelihood with Laplace Smoothing	0.24	0.192	0.8196
Jelinek-Mercer smoothing	0.2	0.178	0.8428
BM25	0.268	0.212	0.9196

Table 2



Graph 2: Precision, Recall for all models for IS4200/CS6200

From the above results of NIST and IS4200/6200 QREL. Following were the observations (Refer Table 1 and Table 2):

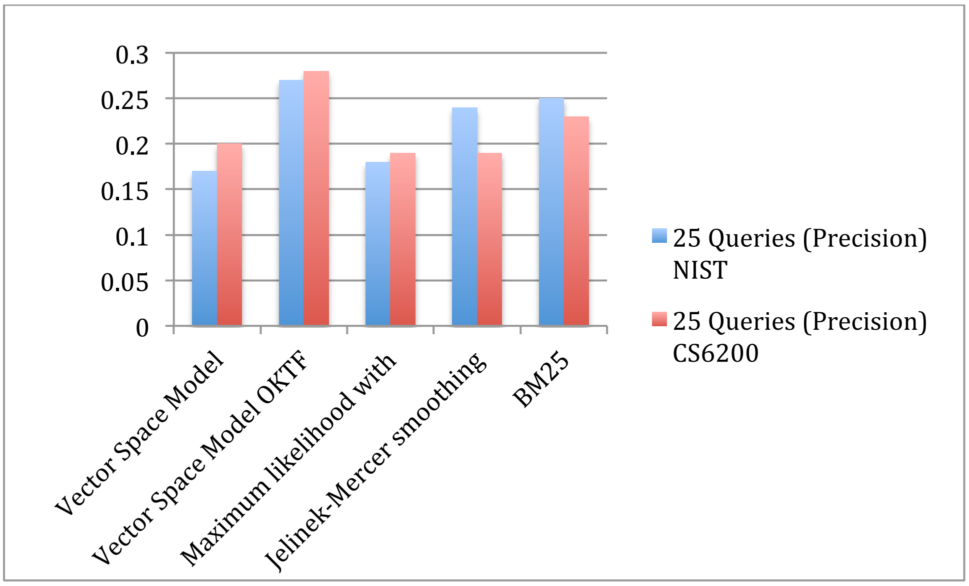
1. No. of relevant documents as per both the models are different. This might have affected the recall. NIST has more number of relevant documents hence over all recall for all models is less than the over all of IS4200/6200 QREL.
2. Precision in both the cases is reducing as number of documents increase. i.e. Precision values reduce at 30 compared to 10. This is because as more and more documents are taken chances of adding an irrelevant document to the retrieved document is set is more. Hence the precision decreases.
3. Recall of both the models is over all on the higher side which means higher percentage of relevant documents were retrieved by the retrieval models implemented.

Empirical Analysis of performances of retrieval models:

Precision :

Model	25 Queries (Precision)	
	NIST	CS6200
Vector Space Model (OKTF)	0.17	0.2
Vector Space Model OKTF with IDF	0.27	0.28
Maximum likelihood with Laplace Smoothing	0.18	0.19
Jelinek-Mercer smoothing	0.2397	0.19
BM25	0.25	0.23

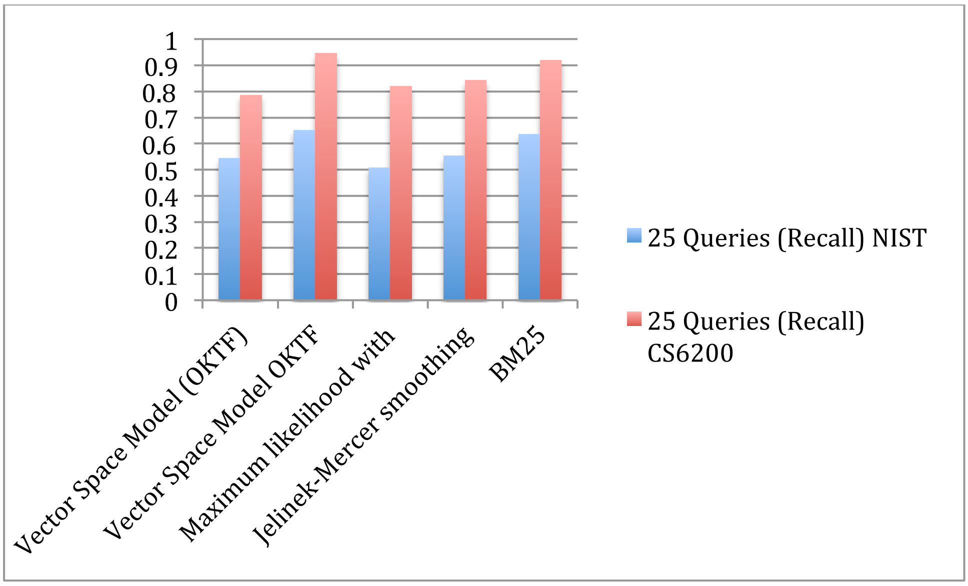
Table 3: Precision values for models in NIST and CS 6200



Graph 3: Precision values for both qrels per model

Model	25 Queries (Recall)	
	NIST	CS6200
Vector Space Model (OKTF)	0.544	0.7857
Vector Space Model OKTF with IDF	0.652	0.9464
Maximum likelihood with Laplace Smoothing	0.5076	0.8196
Jelinek-Mercer smoothing	0.553	0.8428
BM25	0.635	0.9196

Table 4: Recall values for models in NIST and CS 6200



Graph 4: Recall values for both qrels per model

From the above graphs 3 and 4, it is clear that precision and recall values that were generated by all the models on both qrels were highest for Vector Space model with IDF . BM25 also showed pretty good precision and recall values. Precision values for Jelinek-Mercer smoothing were moderate. Vector Space Model And Laplace has very similar precision values.

Both the qrels show similar trends. Hence out of the 5 models implemented Vector Space with IDF and BM25 showed good results.

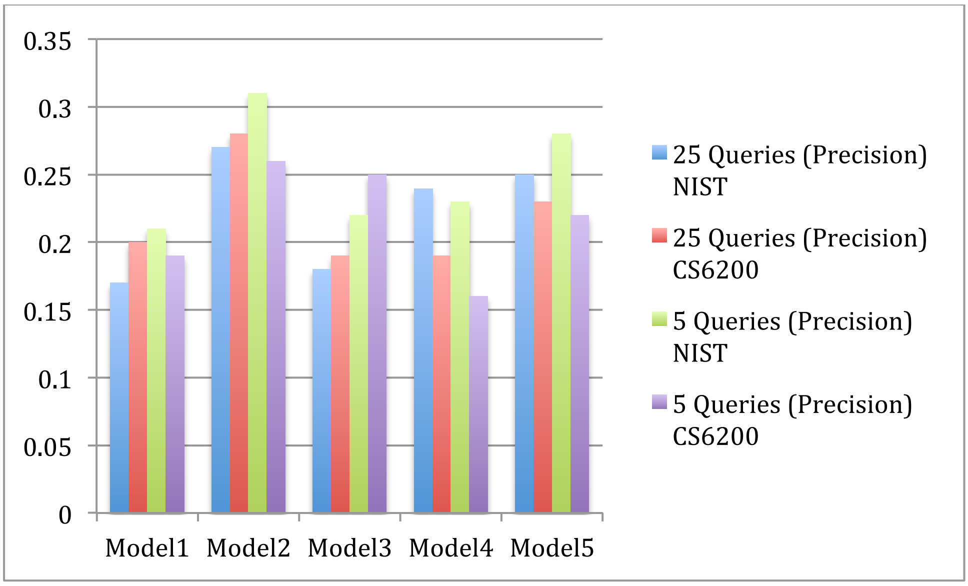
Theoretically addition of IDF drastically would have changed the weights giving better results. Also for BM25 , the binary independence model of relevance and non relevance probabilities would helped in attaining better results. For BM25,the results pretty much depend on the value of k1,k2. If those values are changed we might get much better precisions.

Analysis of models on number of queries:

When 5 models were run on 5 queries ,following results were obtained.

Model	25 Queries (Precision)		5 Queries (Precision)	
	NIST	CS6200	NIST	CS6200
Model1	0.17	0.2	0.21	0.19
Model2	0.27	0.28	0.31	0.26
Model3	0.18	0.19	0.22	0.25
Model4	0.2397	0.19	0.23	0.16
Model5	0.25	0.23	0.28	0.22

Table 5 No. of queries and model performance analysis



Graph 5: No. of queries and model performance analysis

The results on 5 queries were compared to results obtained after running 25 queries. The precision of Vector Space model and BM25 are similar and the values are higher compared to other three models.

Jelinek-Mercer smoothing λ Analysis

Analysis was done for different values of λ. Due to server limitation, code was run 3 times for different values of lambda on 25 queries and following results were obtained.

For 25 queries	
λ	Precision
0.2	0.23
0.5	0.2138
0.9	0.2

This shows that as value of λ. increases, the precision decreases.

Precision of Vector Space model with d=1

For the first vector space model, the retrieval was performed with database d=1 i.e. without removing stop words form the query and following precision was obtained. This precision is too low because of retaining the stop words.

Queryid (Num): 25
Total number of documents over all queries
Retrieved: 25000
Relevant: 1832

Rel_ret: 670
Interpolated Recall - Precision Averages:
at 0.00 0.3846
at 0.10 0.2214
at 0.20 0.1336
at 0.30 0.0872
at 0.40 0.0560
at 0.50 0.0446
at 0.60 0.0268
at 0.70 0.0156
at 0.80 0.0131
at 0.90 0.0085
at 1.00 0.0006
Average precision (non-interpolated) for all rel docs(averaged over queries)
0.0753
Precision:
At 5 docs: 0.2240
At 10 docs: 0.2120
At 15 docs: 0.1813
At 20 docs: 0.1580
At 30 docs: 0.1360
At 100 docs: 0.0892
At 200 docs: 0.0704
At 500 docs: 0.0422
At 1000 docs: 0.0268
R-Precision (precision after R (= num_rel for a query) docs retrieved):
Exact: 0.1166

Some other observations:

When removing the stop words, word "document" was also considered for vector space model .It improved the precision but for the whole project but currently generated outputs are **without** removing document word.
The reason why word document was not removed was ,document word was important as if someone intended to search videos, images then specific kind of results are expected. When "document " is present in the query it would be expected to have documents as the resulting query. Though for Lemur database it will not be relevant, "document" word was preserved in queries.

References :

[1] Slides by Prof. David Smith
[2] Textbook : An Introduction to Information Retrieval, Christopher D. Manning Prabhakar Raghavan Hinrich Schütze
[3] https://www.khanacademy.org/math/probability/random-variables-topic/binomial_distribution/v/binomial-distribution-3
[4] Database Management Systems, 3rd Edition
[5] http://nlp.stanford.edu/IR-book/html/htmledition/using-query-likelihood-language-models-in-ir-1.html
[6] http://en.wikipedia.org/wiki/Binomial_distribution
[7] http://en.wikipedia.org/wiki/Language_model
[8] http://www.youtube.com/watch?v=BFHGIE-nwME
[9] http://nlp.stanford.edu/IR-book/html/htmledition/okapi-bm25-a-non-binary-model-1.html
[10] http://en.wikipedia.org/wiki/Additive_smoothing
[11] http://www.youtube.com/watch?v=ZEkO8QSlynY
[12] http://www.youtube.com/watch?v=BFHGIE-nwME
[13] http://www.youtube.com/watch?v=fvNUUJuFXM0