REPORT

Analysis of TREC_eval result on different Models:

We have implemented the following Similarity models:

1. `LMDirichletSimilarityFactory` Language Model
2. `DefaultSimilarityFactory or VSM`
3. `BM25SimilarityFactory`

1. `LMDirichletSimilarityFactory` Language Model:

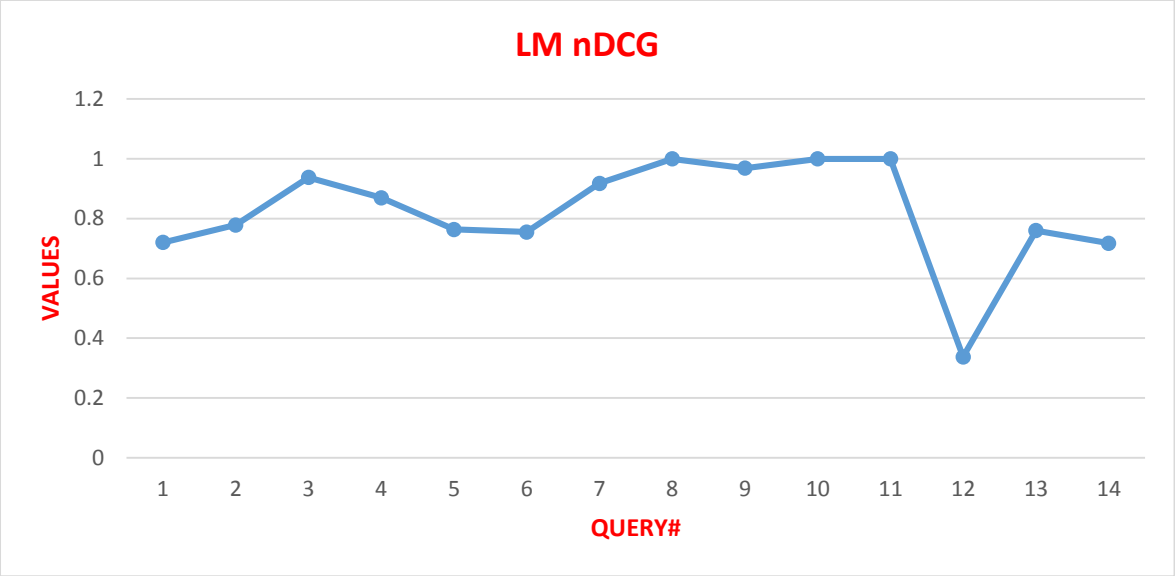
The 14 test queries have been evaluated using TREC_eval and the following measurements have been obtained.

		LM		
Serial #	**NDCG**	**MAP**	**F_0.5**	**BPREF**
1	0.7201	0.3121	0.059	0.5875
2	0.7785	0.4766	0.0796	0.6612
3	0.9379	0.6483	0.0189	0.8667
4	0.8697	0.5434	0.2008	0.5169
5	0.7639	0.5042	0.0095	0.5
6	0.755	0.5034	0.0298	0.4815
7	0.918	0.6429	0.0045	0.5
8	1	1	0.0165	1
9	0.9688	1	1	1
10	1	1	0.027	1
11	1	1	0.75	1
12	0.3376	0.1118	0.0194	0.568
13	0.7597	0.1998	0.1524	0.261
14	0.7174	0.5838	0.12	0.6914
Avg. Values	0.823329	0.609	0.1777	0.6882

1. **nDCG** : **Normalized discounted cumulative gain (NDCG)** measures the performance of a recommendation system based on the graded relevance of the recommended entities. It varies from 0.0 to 1.0, with 1.0 representing the ideal ranking of the entities. This metric is commonly used in information retrieval and to evaluate the performance of web search engines.

$$nDCG_k = \frac{DCG_k}{IDCG_k}$$

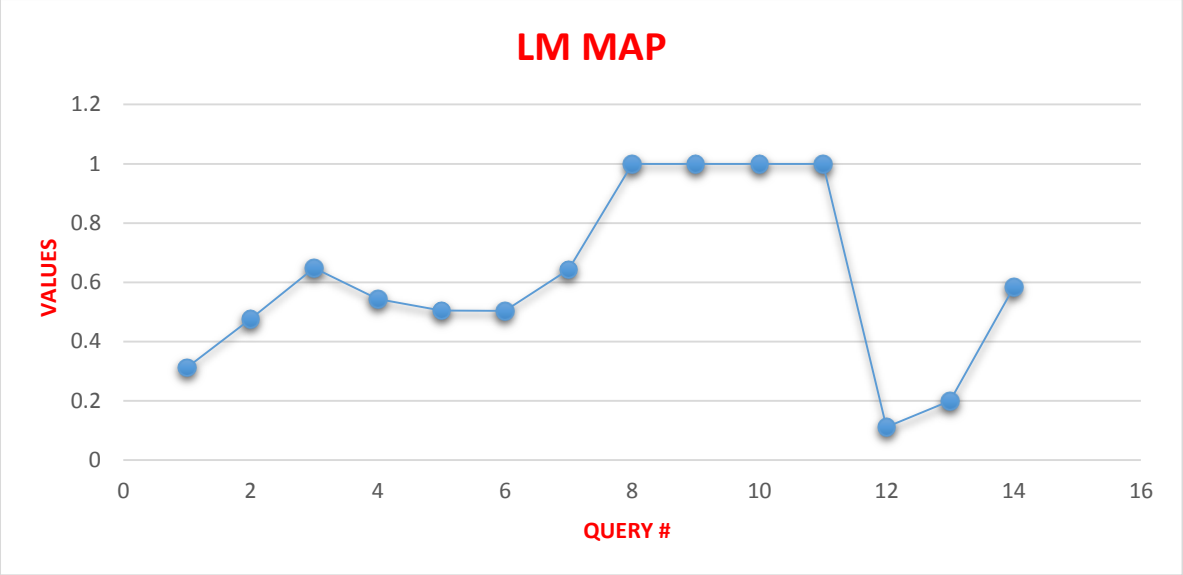
$IDCG_k$ is the maximum possible (ideal) DCG for a given set of queries, documents, and relevances.
k - The maximum number of entities that can be recommended.



Observation:

We find that the average value of this measurement comes to around 0.8233 which nears the ideal ranking of the entities. This proves that highly relevant documents are more useful when appearing earlier in a search engine result list and have higher ranks.

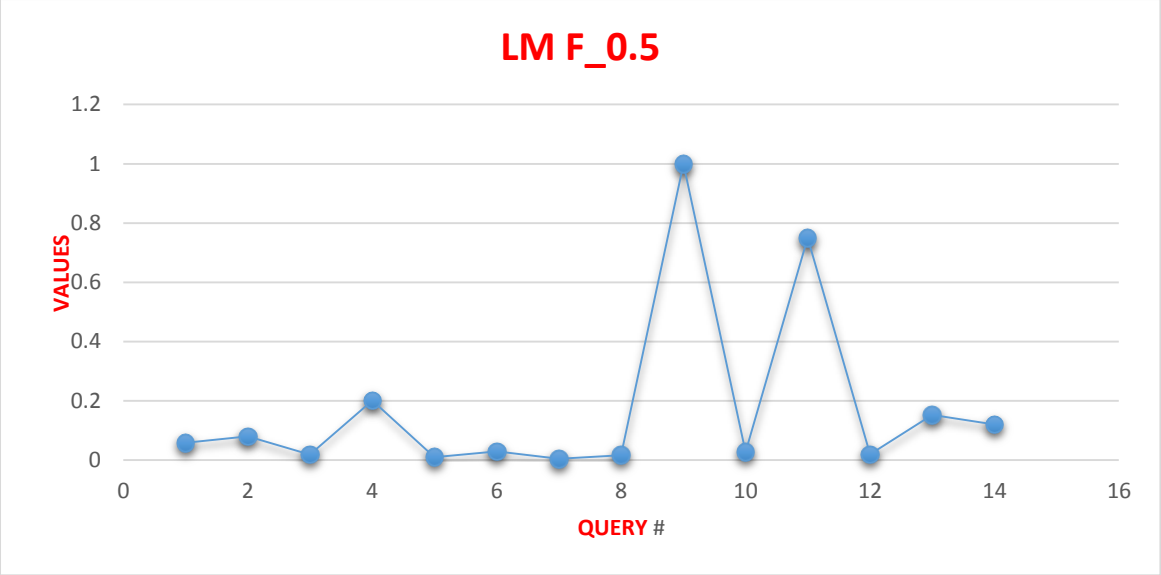
2. **MAP**: Most standard among the TREC community is *Mean Average Precision* (MAP), which provides a single-figure measure of quality across recall levels. Among evaluation measures, MAP has been shown to have especially good discrimination and stability.
   While precision is the fraction of the retrieved documents that are relevant, average precision is a single value obtained by averaging the precision values at each new relevant document observed.



Observation: The average value of this measurement comes to 0.609 which indicates better performance by providing high precision results at the top of the ranked list.

3. **F-MEASURE**: It considers both the precision *p* and the recall *r* of the test to compute the score: *p* is the number of correct positive results divided by the number of all positive results, and *r* is the number of correct positive results divided by the number of positive results that should have been returned. The F_1 score can be interpreted as a weighted average of the precision and recall, where an F_1 score reaches its best value at 1 and worst at 0.
   The traditional F-measure or balanced F-score (**F_1 score**) is the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

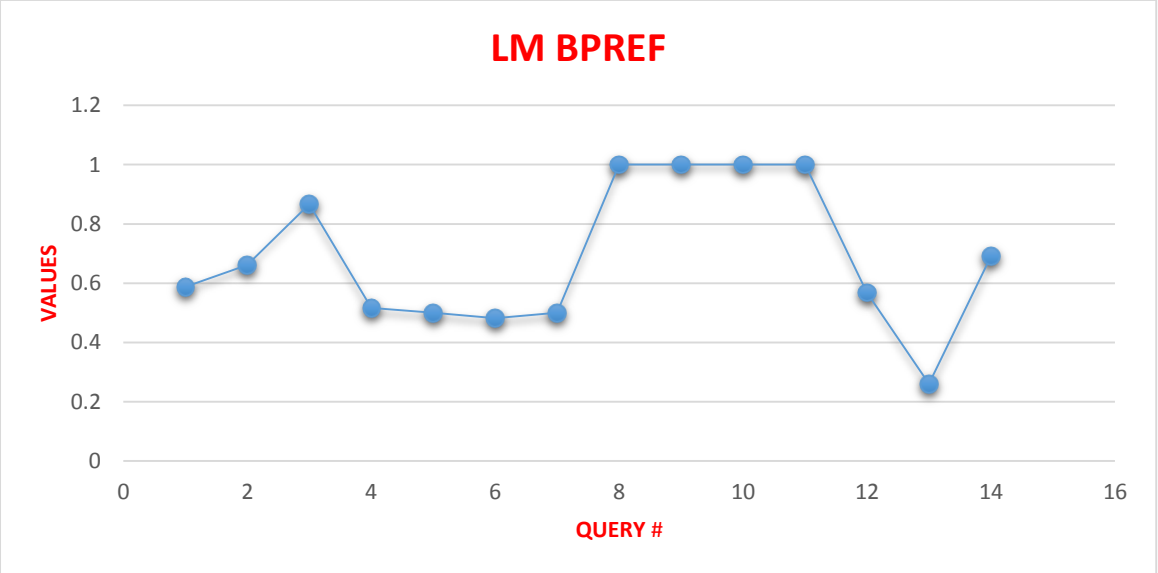


Observation: We find that the F-measure is low for most of the queries after TREC_EVAL. Although, we have high F-measure score for Query# 9 and Query# 11.

4. **BPREF:** which stands for **binary preference**. It computes a preference of whether judged relevant documents are retrieved ahead of judged non-relevant documents. It is defined as

$$bpref = \frac{1}{R}\sum_{r}\left(1 - \frac{|n \text{ ranked higher than } r|}{min(R, N)}\right)$$

R: number of judged relevant documents
N: number of judged non-relevant documents
r: relevant document retrieved
n: member of the first R judged non-relevant documents retrieved



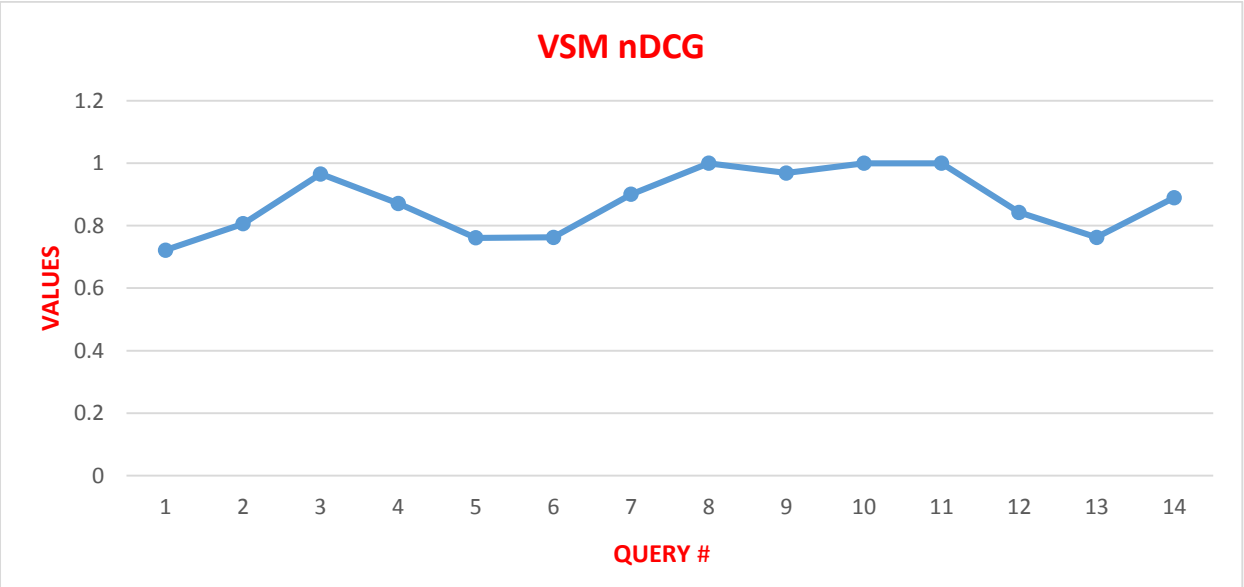
Observation: As we can see, bpref allows us to simply look at how known relevant and non-relevant documents are ranked rather than expecting to know all the relevant documents in the collection. The average comes to about 0.6882 from running Trec_Eval on 14 given queries.

2. DefaultSimilarityFactory or VSM

Next, we have implemented the VSM Model which is actually the default similarity model:

		VSM		
Serial #	**NDCG**	**MAP**	**F_0.5**	**BPREF**
1	0.7219	0.3167	0.059	0.5875
2	0.8063	0.5502	0.0796	0.6612
3	0.9655	0.7526	0.0189	0.9167
4	0.8714	0.5514	0.2008	0.5169
5	0.7613	0.5029	0.0095	0.5
6	0.7627	0.5224	0.0298	0.5309
7	0.9007	0.5833	0.0045	0.5
8	1	1	0.0165	1
9	0.9688	1	1	1
10	1	1	0.027	1
11	1	1	0.75	1
12	0.8424	0.3794	0.027	0.6982
13	0.7628	0.2047	0.1524	0.2688
14	0.8897	0.6962	0.12	0.7531
Avg. Values	0.87525	0.647129	0.178214	0.709521

1. **nDCG**:

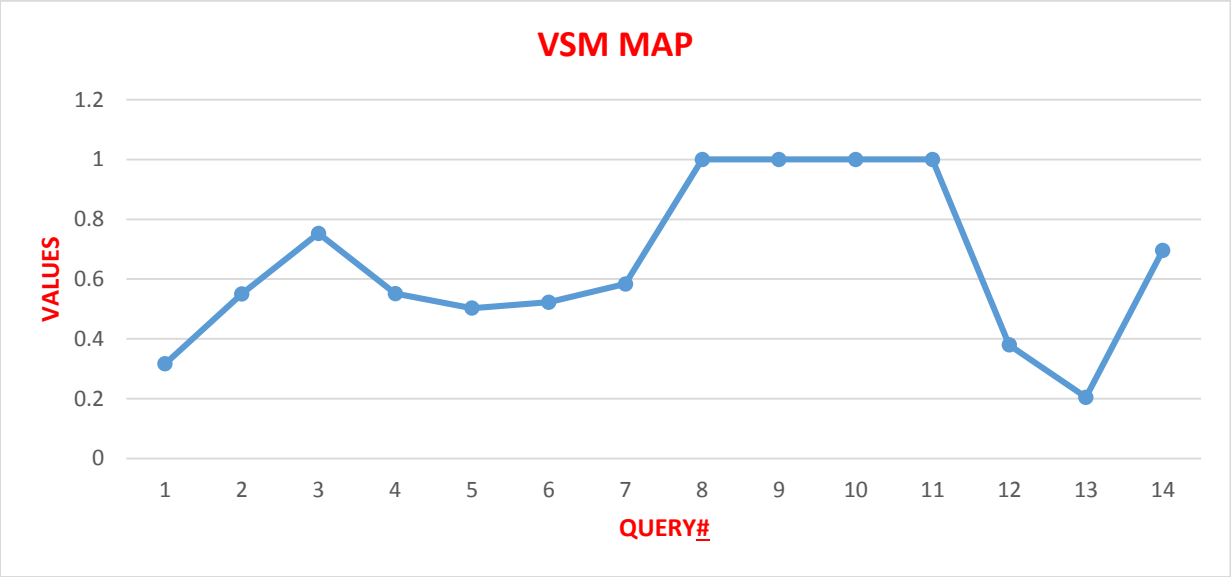


Observation: The average score of nDCG of the 14 queries comes to 0.87525 which performs better than LM.

So,

nDCG(VSM) > nDCG(LM)

2. **MAP:**

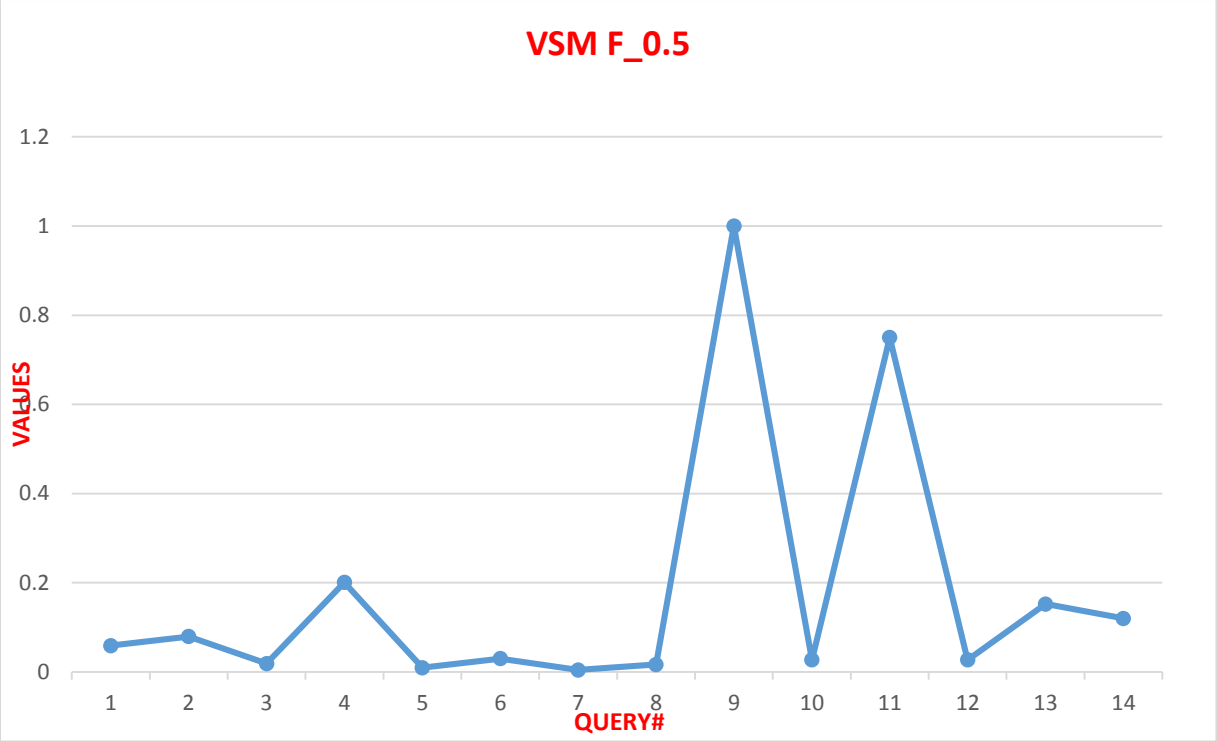


Observation: The average score of MAP of the 14 queries comes to 0.647129 which performs better than LM.

So,

Map(VSM) > Map(LM)

3. **F Measure:**

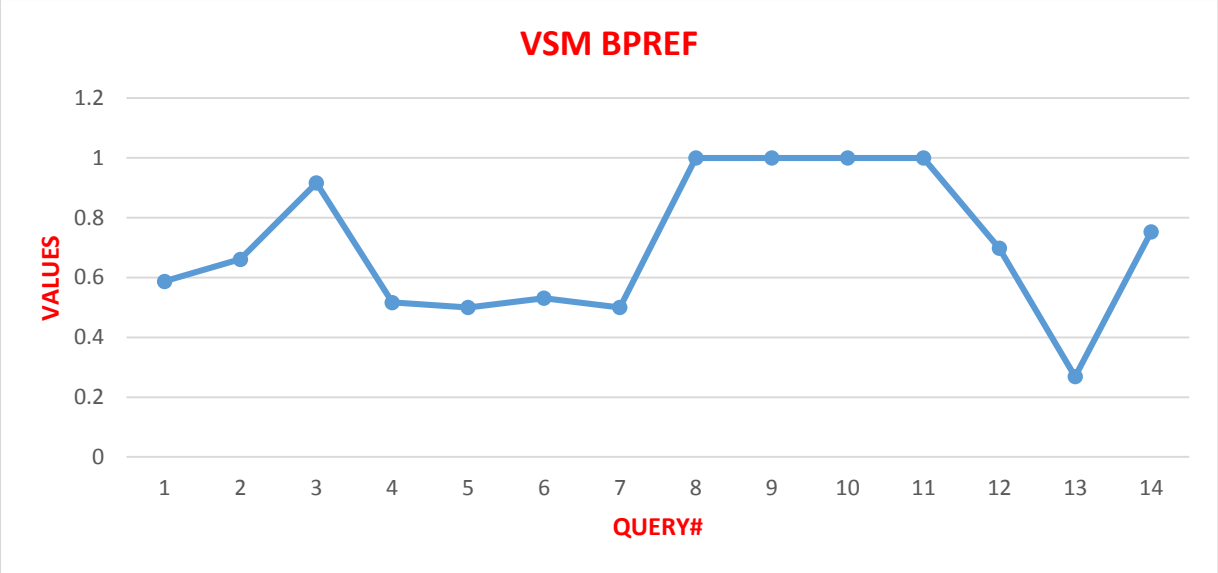


Observation: The average score of F_0.5 of the 14 queries comes to 0.178214 which performs better than LM.

So,

F_0.5(VSM) > F_0.5(LM)

4. **BPREF:**



Observation: The average score of MAP of the 14 queries comes to 0.709521 which performs better than LM.

So,

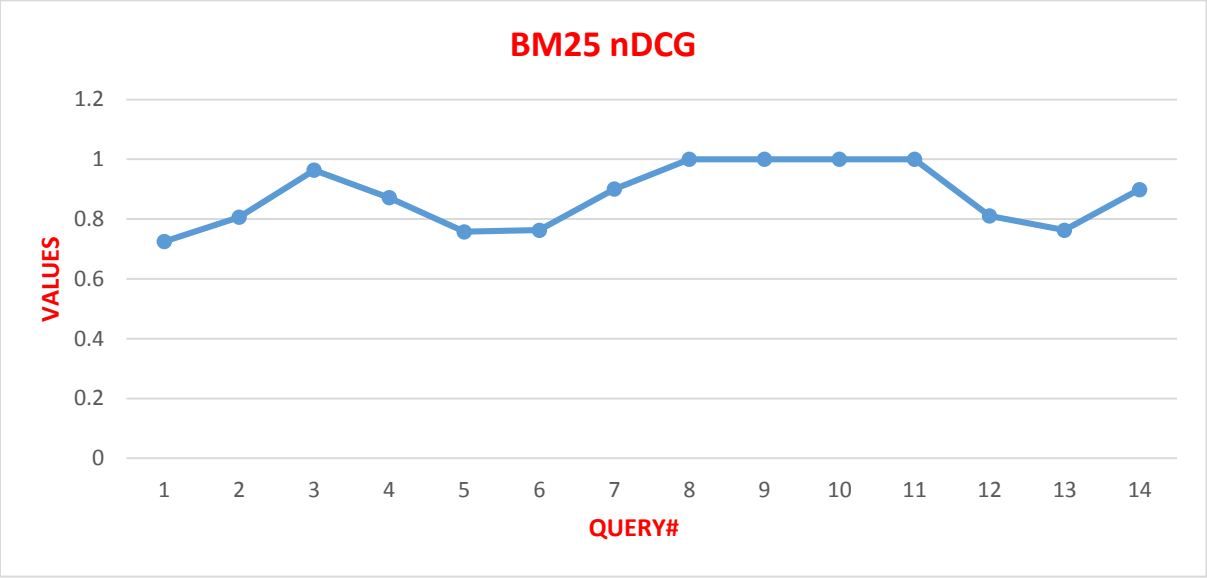
bpref(VSM) > bpref (LM)

3. BM25SimilarityFactory

Next, we have implemented the BM25SimilarityFactory Model:

		BM		
Serial #	**NDCG**	**MAP**	**F_0.5**	**BPREF**
1	0.7248	0.3218	0.059	0.5875
2	0.8063	0.5502	0.0796	0.6612
3	0.9635	0.7387	0.0189	0.9167
4	0.8714	0.5514	0.2008	0.5169
5	0.7577	0.5016	0.0095	0.5
6	0.7627	0.5224	0.0298	0.5309
7	0.9007	0.5833	0.0045	0.5
8	1	1	0.0165	1
9	1	1	1	1
10	1	1	0.027	1
11	1	1	0.75	1
12	0.8106	0.3262	0.027	0.5562
13	0.7628	0.2047	0.1524	0.2688
14	0.8991	0.7083	0.12	0.7407
Avg. Values	0.875686	0.643471	0.178214	0.698493

1. **nDCG:**

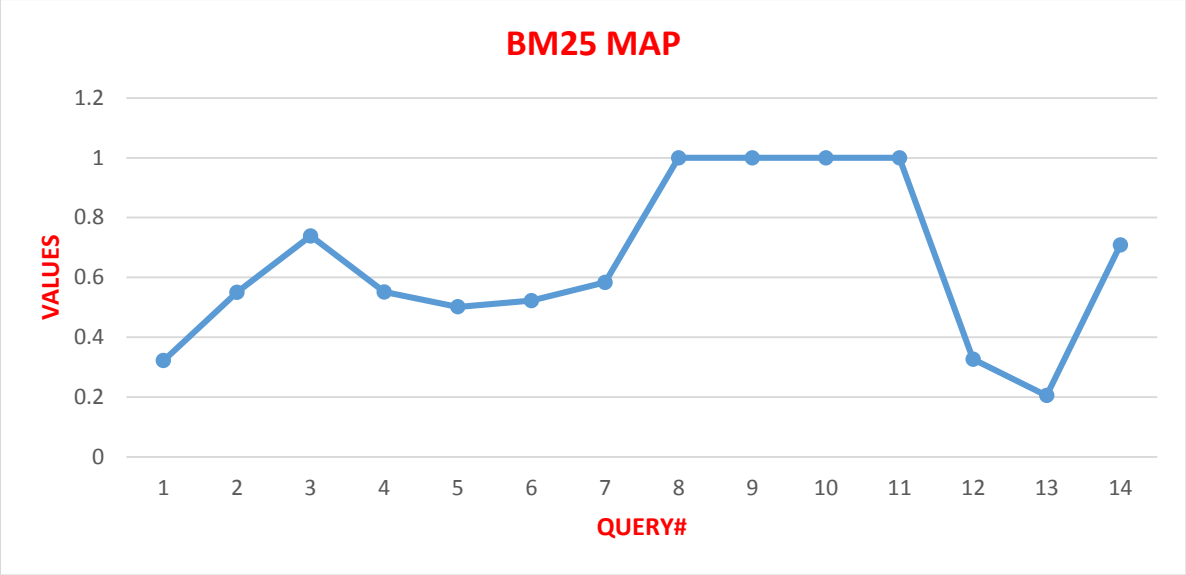


Observation: The average score of nDCG of the 14 queries comes to 0.875686 which performs better than LM and VSM.

So,

nDCG(BM) > nDCG(VSM) > nDCG(LM)

2. **MAP:**

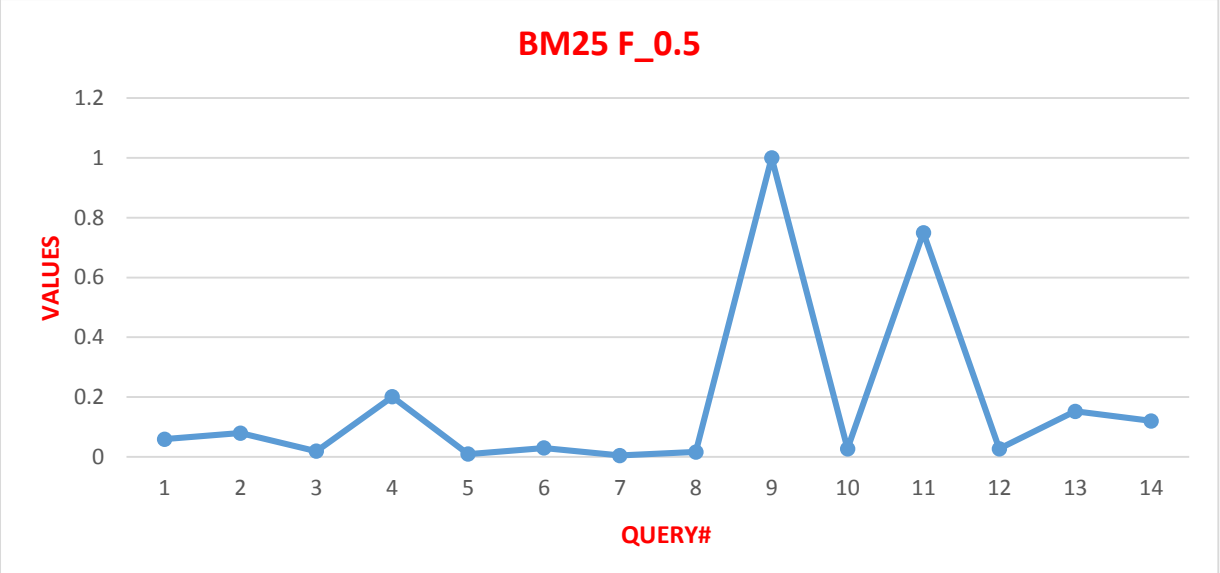


Observation: The average score of MAP of the 14 queries comes to 0.643471 which performs better than LM but less than VSM.

So,

Map(VSM) > Map(BM) > Map(LM)

3. **F0.5**

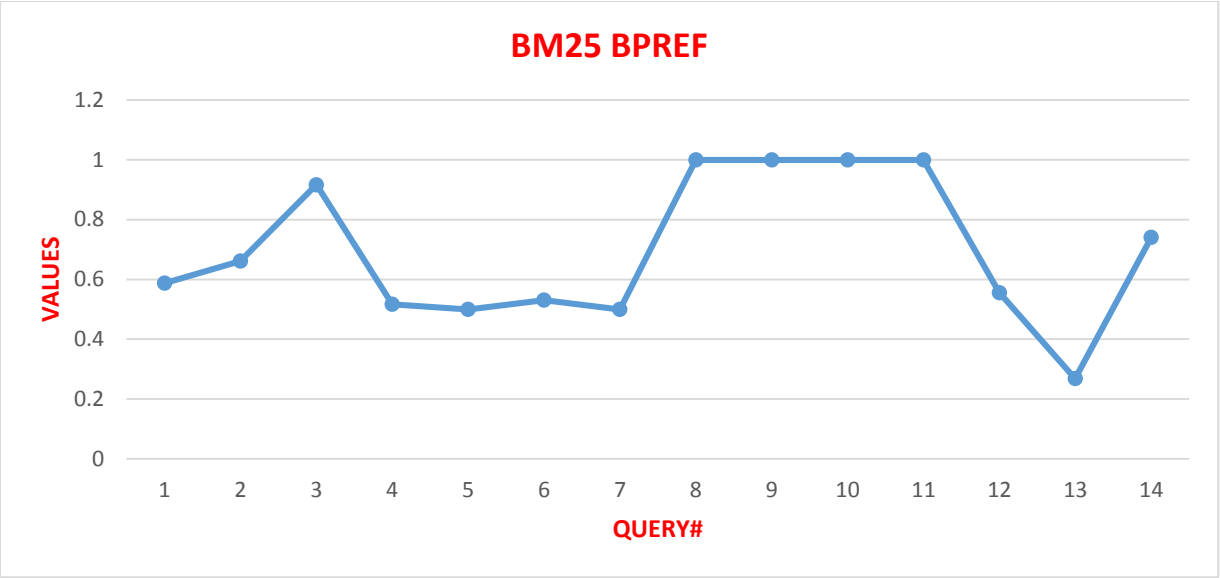


Observation: The average score of F_0.5 of the 14 queries comes to 0.178214 which is same as VSM but better than LM.

So,

$$F\_0.5(BM) = F\_0.5(VSM) > F\_0.5(LM)$$

4. **BPREF:**



Observation: The average score of BPREF of the 14 queries comes to 0.698493 which performs better than LM but less than VSM.

So,

$$bpref(VSM) > bpref(BM25) > bpref\ (LM)$$

Better Overall Performance:

The DefaultSimilarity Model or the VSM Model performs better overall as when the scores are compared for different measurements, VSM ranks higher for the four measures among the other Similarity Models. The following table proves the analysis.

	NDCG	MAP	F_0.5	BPREF
VSM	0.8753	0.6471	0.1782	0.7095
BM25	0.8756	0.6434	0.1782	0.6984
LM	0.8233	0.609	0.1777	0.6882

What have you done to improve the performance?

Answer:

1. We have implemented **Edismax or Extended Dismax** Query Parser for querying words across multiple fields with different boosts based on the significance of each field.

2. We have used Query Boosters by tweaking various parameters such as 'qf' and 'bq' which boosts the specified fields. For eg. For text_en^2 ; text_en ^ 0.2 text_ru ^ 2 text_de ^ 0.4.

3. We have converted the original query into different languages such as English, Russian and German to further improve the results of the system.

4. In order to further improve the system, we tried using Query Filter such as **ShingleFilterFactory** but we got results which were unsatisfactory and performance of the system dropped considerably. Hence we decided not to use the Query Filter.

5. We have also used "solr.UAX29URLEmailTokenizerFactory" to further improve the quality of the system.

6. We also added synonyms of commonly used words manually into 'Synonyms.txt'. For eg. For "refugee" we added words such as "alien", "exile", "immigrant", "foreigner". For "crisis" we added "disaster", "catastrophe", "impasse".

7. We also modified the existing Stopwords.txt for English, Russian and German to remove words such as "rt", "http", "https" to improve our search results.

8. **Fuzzy searching** was also used but it only increased overall number of documents that were retrieved without a significant increase in evaluation measures.

Performance Change on using Different Setups

Before implementing Query Boosting, Query Parsing and Language Conversion, the values for test **Query #1** were:

Serial #	NDCG	MAP	F_0.5	BPREF
1	0.1443	0.0386	0.1184	0.3

This also includes the **ShingleFilterFactory** has also been used to filter the query.

The values obtained are same for all the models, i.e, LM, BM25 and VSM.

After implementing Query Boosting (**qf**), Query Parsing (**edismax**) and Language Conversion(en<->ru<->de), the values for test query #1 are given as follows:

QUERY #1 for BM25 Model	NDCG	MAP	F_0.5	BPREF
Plain Query + ShingleFilterFactory	0.1443	0.0386	0.1184	0.3
Query Boosting + Query Parser + Language Conversion	0.7248	0.3218	0.059	0.5875

QUERY #1 for VSM Model	NDCG	MAP	F_0.5	BPREF
Plain Query + ShingleFilterFactory	0.1443	0.0386	0.1184	0.3
Query Boosting + Query Parser + Language Conversion	0.7219	0.3167	0.059	0.5875

QUERY #1 for LM Model	NDCG	MAP	F_0.5	BPREF
Plain Query + ShingleFilterFactory	0.1443	0.0386	0.1184	0.3
Query Boosting + Query Parser + Language Conversion	0.7201	0.3121	0.059	0.5875

We see that there has been a huge difference in values of nDCG, MAP, F_0.5 and BPREF for all the models.

Similarly, we try to improve **Query #4** :

Before it was:

Serial #	NDCG	MAP	F_0.5	BPREF
4	0.3776	0.0943	0.3548	0.4783

After implementing Query Boosting (qf), Query Parsing (edismax) and Language Conversion(en<->ru<->de), we find that the results have improved leaps and bounds.

QUERY #4 for BM25 Model	NDCG	MAP	F_0.5	BPREF
Plain Query + ShingleFilterFactory	0.3776	0.0943	0.3548	0.4783
Query Boosting + Query Parser + Language Conversion	0.8714	0.5514	0.2008	0.5169
QUERY #4 for VSM Model	NDCG	MAP	F_0.5	BPREF
Plain Query + ShingleFilterFactory	0.3776	0.0943	0.3548	0.4783
Query Boosting + Query Parser + Language Conversion	0.8714	0.5514	0.2008	0.5169
QUERY #4 for LM Model	NDCG	MAP	F_0.5	BPREF
Plain Query + ShingleFilterFactory	0.3776	0.0943	0.3548	0.4783
Query Boosting + Query Parser + Language Conversion	0.8697	0.5434	0.2008	0.5169

Similarly, we try for **Query# 6:**

Before the values were coming :

Serial #	NDCG	MAP	F_0.5	BPREF
6	0.0579	0.0032	0.0323	0.1111

After using Query Boosting (qf), Query Parsing (edismax) and Language Conversion(en<->ru<->de), it comes to:

QUERY #6 for BM25 Model	NDCG	MAP	F_0.5	BPREF
Plain Query + ShingleFilterFactory	0.3776	0.0943	0.3548	0.4783
Query Boosting + Query Parser + Language Conversion	0.8714	0.5514	0.2008	0.5169
QUERY #6 for VSM Model	NDCG	MAP	F_0.5	BPREF
Plain Query + ShingleFilterFactory	0.3776	0.0943	0.3548	0.4783
Query Boosting + Query Parser + Language Conversion	0.8714	0.5514	0.2008	0.5169
QUERY #6 for LM Model	NDCG	MAP	F_0.5	BPREF
Plain Query + ShingleFilterFactory	0.3776	0.0943	0.3548	0.4783
Query Boosting + Query Parser + Language Conversion	0.8697	0.5434	0.2008	0.5169

The Measure that gives a better evaluation of the system:

Although, MAP is considered to be a good measure for evaluation of a search system. But any search system should fulfil a major requirement that relevant results should appear on the top, i.e. they must rank high than non-relevant documents. This is because, if relevant document is ranked lower in the retrieved set of results then it's a possibility that users may not be interested to look for other results as he found most of them to be non-relevant.

The nDCG system penalizes if some relevant document is ranked lower in the result set. Hence, we consider it to be the best measure for search system used by a human.

Therefore, we believe nDCG gives a better evaluation of the system.