CSE 535 PROJECT PART B

GROUP: PROJECT 404

Member Details:

Arundhati Rao (50169206)

Cuurie Athiyaman (50170367)

Sukanya Moorthy (50170061)

SUMMARY

VSM:

Efficient Configuration:

The most efficient tweaking for VSM was when the default Lucene Query Parser was replaced with disMax Query Parser and max rows=1000 was used in the query.

Metrics	Value for disMax, 1000 rows
Ndcg	0.9134
Bpref	0.7558
F0.5	0.1834
Map	0.7381

BM25

Efficient Configuration:

1)Using URLTokenizer instead of standard tokenizer

2)Parameter Values:

```
<similarity class="solr.BM25SimilarityFactory">
  <float name="k1">1.4</float>
  <float name="b">0.5</float>
</similarity>
```

3)Using dismax query with equal weightage to all text fields

inurl='http://uskk3b0f361e.sukanyamoorthy.koding.io:8983/solr/bm25/select?q='…..'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&defType=dismax&qf=text_en+text_ru+text_de&wt=json&indent=true&rows=10000'

Ndcg	0.9041
Bpref	0.7462
Map	0.7387
F.0.5	0.1834

LM

The performance of the system is optimized when query is optimized using boosting and language translation in a LMDirichlet Similarity model with a smoothing parameter of mu=50. The best measure for the language model is the ndcg measure because language model as such relies a lot on the frequency of the query terms appearing in the documents.When the normalization factor in ndcg is applied, the measure becomes efficient than other measures, which are based on precision.

When the given query input is translated to the corresponding languages, a higher boosting is given to the corresponding language fields, which result in the following measures:

ndcg	0.9061
set_F.0.5	0.2331
map	0.7153
bpref	0.7327

As per analysis VSM model came out to the best with ndcg and bpref being higher than rest of the models but precision being low 0.1834. Higher precision was observed in Language model.

As the system is ranked retrieval model ndcg should be the best measure to evaluate the system . The reason being user might want to see higher ranked results first for example search engine , but otherwise bpref should be the best metric to evaluate the reason being qrel containing lot of unjudged query ,doc pair . In such a system it is also important to see if the docs judged are coming faster than irrelevant docs.

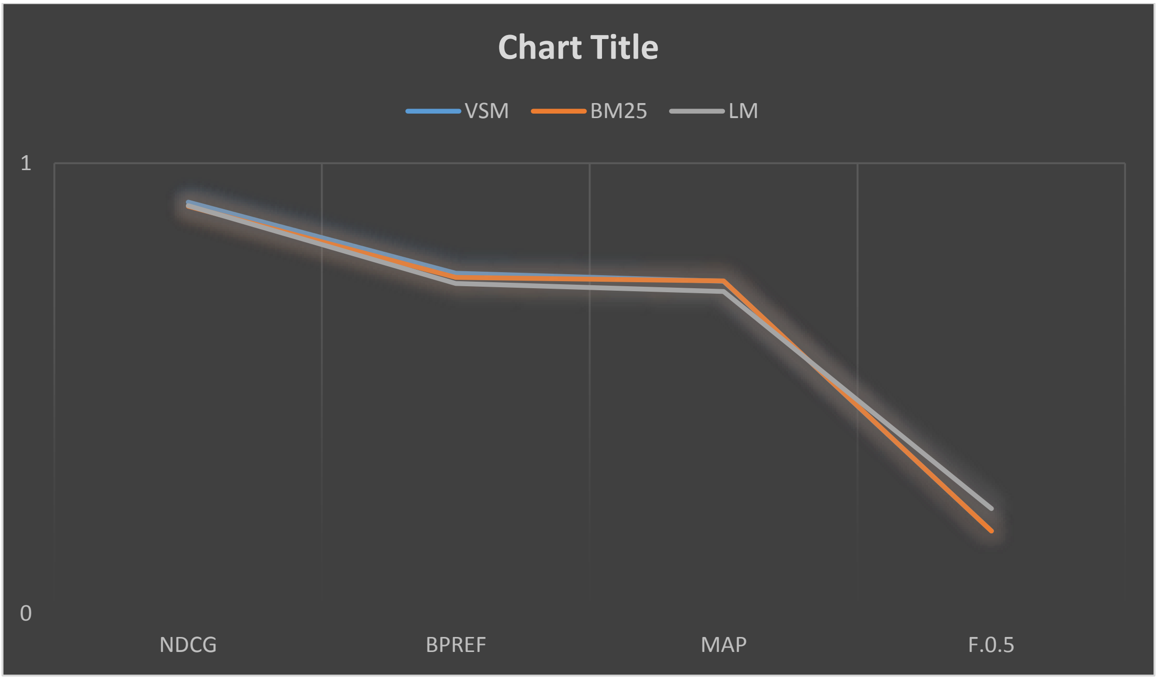
Metrics	**VSM Model**
Ndcg	0.9134
Bpref	0.7558
F0.5	0.1834
Map	0.7381

Details:

In every model we have tried various configuration and query changes as mentioned below in model specific details

In every model we have tried various configuration and specified the reason for performance change in model specific details below

Graph:



SUMMARY

1. Analyze TREC_eval result on different models. Which one performs better overall and why?

Model Results:

VSM:

Efficient Configuration:

The most efficient tweaking for VSM was when the default Lucene Query Parser was replaced with disMax Query Parser and max rows=1000 was used in the query.

Metrics	Value for disMax, 1000 rows
Ndcg	0.9134
Bpref	0.7558
F0.5	0.1834
Map	0.7381

BM25

Efficient Configuration:

1)Using URLTokenizer instead of standard tokenizer

2)Parameter Values:

```
<similarity class="solr.BM25SimilarityFactory">
  <float name="k1">1.4</float>
  <float name="b">0.5</float>
</similarity>
```

3)Using dismax query with equal weightage to all text fields

inurl='http://uskk3b0f361e.sukanyamoorthy.koding.io:8983/solr/bm25/select?q='.....'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&defType=dismax&qf=text_en+text_ru+text_de&wt=json&indent=true&rows=10000'

Ndcg	0.9041
Bpref	0.7462
Map	0.7387
F.0.5	0.1834

LM

The performance of the system is optimized when query is optimized using boosting and language translation in a LMDirichlet Similarity model with a smoothing parameter of mu=50. The best measure for the language model is the ndcg measure because language model as such relies a lot on the frequency of the query terms appearing in the documents.When the normalization factor in ndcg is applied, the measure becomes efficient than other measures, which are based on precision.

When the given query input is translated to the corresponding languages, a higher boosting is given to the corresponding language fields, which result in the following measures:

ndcg	0.9061
set_F.0.5	0.2331
map	0.7153
bpref	0.7327

As per analysis VSM model came out to the best with ndcg and bpref being higher than rest of the models but precision being low 0.1834. Higher precision was observed in Language model.

As the system is ranked retrieval model ndcg should be the best measure to evaluate the system . The reason being user might want to see higher ranked results first for example search engine , but otherwise bpref should be the best metric to evaluate the reason being qrel containing lot of unjudged query ,doc pair . In such a system it is also important to see if the docs judged are coming faster than irrelevant docs.

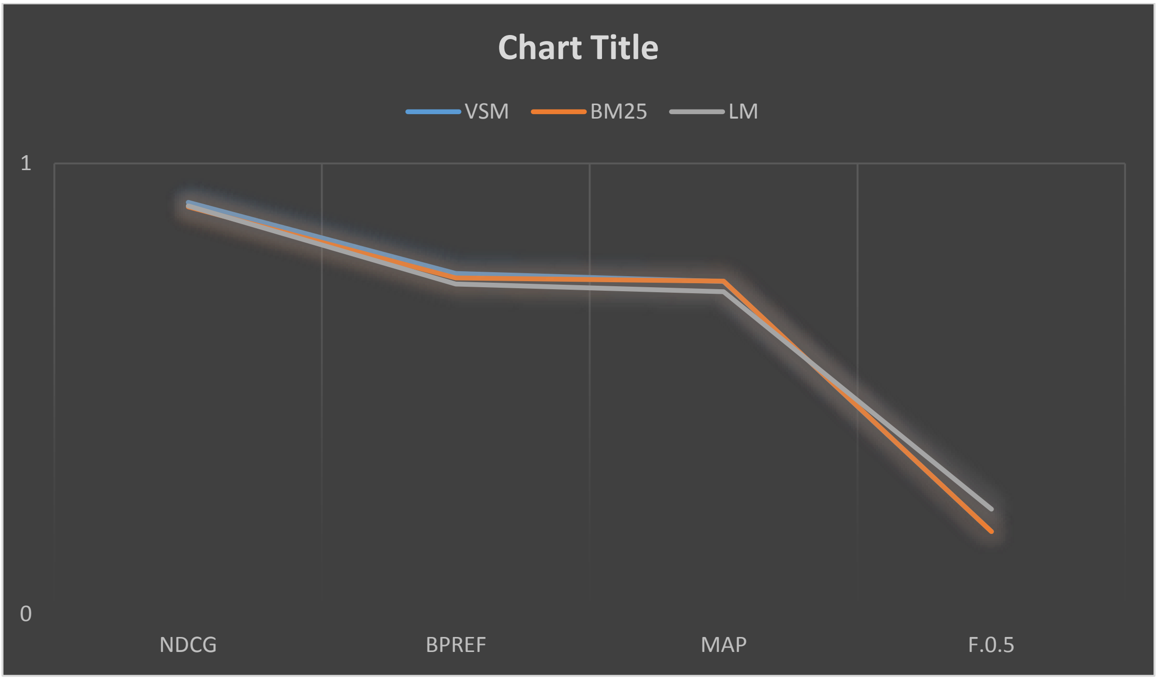
Metrics	**VSM Model**
Ndcg	0.9134
Bpref	0.7558
F0.5	0.1834
Map	0.7381

Details:

In every model we have tried various configuration and query changes as mentioned below in model specific details

In every model we have tried various configuration and specified the reason for performance change in model specific details below

Graph:



BM25

Summary

For the various test cases tried out the best measure was observed for the following configuration:

Using URLTokenizer instead of standard tokenizer

Parameter Values:

```
<similarity class="solr.BM25SimilarityFactory">
  <float name="k1">1.4</float>
  <float name="b">0.5</float>
</similarity>
```

Using dismax query with equal weightage to all text fields

inurl='http://uskk3b0f361e.sukanyamoorthy.koding.io:8983/solr/bm25/select?q=’…..’&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&defType=dismax&qf=text_en+text_ru+text_de&wt=json&indent=true&rows=10000'

Ndcg	0.9041
Bpref	0.7462
Map	0.7387
F.0.5	0.1834

Though ndcg is higher than than bpref according to the overall analysis bpref is the best measure to tell about the system.

Reason: In the qrel.txt there were many docid query pairs which were not judged and had a relevance of -1. So in such a system it is important to understand how quick the judged documents come than irrelevant documents. If the user is very particular about ranking of documents – for example in search engines then it is important to look for ndcg than bpref

1. Default parameters and max rows=10

Default Parameter
<float name="k1">1.2</float>
<float name="b">0.75</float>

Metrics	Value
Ndcg	0.7964
Bpref	0.6091
F0.5	0.6085
Map	0.6101
Num_ret	138
Num_rel	210
Num_rel_retrieved	92

As we restrict the rows to 10 the precision is very good and F measure is around 0.6 but rest of the documents are not very good as there are many relevant documents even with higher relevance value were not retrieved

2. Increasing number of rows from 10 to 1000

Metric	No of Rows -10	No of Rows -1000
Ndcg	0.7964	0.8996
bpref	0.6091	0.7352
Map	0.6101	0.7259
F0.5	0.6085	0.1834

Observation: By increasing the number of rows fetched from 10 to 1000 overall performance of all the metrics increased because recall increased .Due to increase in recall .F measure came down

3. Increasing number of rows from 1000 to 2000

Metric	No of Rows -1000	No of Rows -2000
ndcg	0.8996	0.8996
bpref	0.7352	0.7352
Map	0.7259	0.7259
F0.5	0.1834	0.1834

Observation: By increasing the number of rows fetched from 1000 to 2000 no metric value increase was observed because recall of 100% was achieved between 1000 rows

4. Query by Query analysis based on results retrieved from Test 1

Analysis on query 001:

Out of 20 relevant docs only 7 were relevant. The documents which were relevant were with high relevance value

653278536707506176	1
653278466788487168	3
653278355677184000	3
653278331278913536	3
653941482882134016	3
653941291722645504	1

Certain documents were retrieved even if they had relevance 1 over other relevant documents with score 1 **because of shorter lengths**

As no translation is applied documents in other languages were not retrieved even if they had higher relevance (there was only 1 such document)

Observation: query 001 fetched docs with high relevance but wasn't fetching queries in different languages or fields with longer lengths

Analysis on query 002:

Out of 20 relevant docs only 7 were relevant. The documents which were relevant were with high relevance value

653941469523214336	3
653941400904536064	2
653941391769210880	2
653941338560438272	2
653941282583257088	1
654279474809319425	1
654279412926517249	3
654279396598116352	1

As no translation is applied documents in other languages were not retrieved even if they had higher relevance

Certain documents were not retrieved because relevance was judged based on hashtags as per observation

Observation:

Queries evaluated based on hashtag weren't retrieved and fields in different language to query weren't retrieved (2 such docs)

Query 003 Analysis:

653941336299606016	3
651023411322462208	3
652488980856995840	3
647458516115025921	1
647458476734595074	3
647458045761601536	3
648234039841845248	3

Observation: terms which had repetitive words were not fetched even though it was relevant and fields in different language to query weren't retrieved (2 such docs) and certain fields with longer lengths weren't fetched

Query 004 analysis:

647450320449241088	3
647448990443827200	3
647448171325620224	3
647440709788807168	3
647440220451905536	3
647439236203982849	3
647437149781016576	3
647436890942087168	3
647435786560929792	3
647435592133943296	3

Observation: all documents with high relevance (3) were retrieved. This had many relevant docs in English which were not retrieved as query was in German. So either translation or giving weightage to English docs too even though searched in German has to be taken care. In certain docs hyphen was present such as merkel-plan (such relevant docs were not retrieved). Relevant docs had repetitive terms and longer lengths too, so k1 and b may have to be adjusted accordingly. One or two queries hashtag preference was present

Query 005

653278261678579712	3
653274267207999488	3

Observation: all documents with high relevance (3) were retrieved. This had 2 relevant docs in English which were not retrieved as query was in Russian. Translation and hashtag based field search should be applied to text_en when search in Russian.

Query 006

653278437910573057	3
653278312249225216	1
653941412421963778	3
653941305551155200	3
653941285045276672	2

Observation: all documents with high relevance (3,2) were retrieved. This had 3 relevant docs in german which were not retrieved as query was in english. Translation and hashtag based field search should be applied

Query 007

653720091985469440	3
652113514954420224	1

Observation: Both the documents were retrieved

Query 008

653278366427181056 3
653278352816635904 3
653278329521504256 3
653941412636008449 3
653941409829883904 3
653941264690249728 3
654279404542140416 3
654279266780188672 3

Observation: all documents with high relevance (3) were retrieved. The query was basically RT @Free_Media_Hub ,so as no underscore removing tokenizer was implemented the query fetched relevant results

Query 009

Observation: all documents with relevant documents were retrieved.

Query 010

653278261678579712	3
653274267207999488	3

Observation: all documents with high relevance (3) were retrieved. This had 2 relevant docs in English which were not retrieved as query was in Russian. Translation and hashtag based field search should be applied to text_en when search in Russian.

Query 011

653278261678579712	3
653274267207999488	3

Observation: all documents with high relevance (3) were retrieved. This had 2 relevant docs in English which were not retrieved as query was in Russian. Translation and hashtag based field search should be applied to text_en when search in Russian.

5. Query Parser – changing default field to AND

Observation: This decreased the overall scores of all metrics drastically. As we restrict our search to complete phrases by giving 'and' it resulted in fetching very few documents

Metrics	Value
Ndcg	0.2963
Bpref	0.2784
F0.5	0.3099
Map	0.2784

6. Query Parser – Proximity Searches

Proximity ~5

Metrics	Previous	Current
ndcg	0.8996	0.8689
bpref	0.7352	0.7312
F0.5	0.1834	0.1249
Map	0.7254	0.6603

As there not many documents with proximity ~ 5 , it ended up fetching many irrelevant documents and documents with higher rank were coming at a later stage . hence overall there was decrease in all the metrics

7. Query Parser – Query Boosting

As per each query observation if query is in English then English docs have higher weightage over rest of the languages

If query weightage in German, German doc will have highest weightage, next highest should be English

If query weightage in Russian, Russian doc will have highest weightage, next highest should be English

German Query: qf =text_en^0.25+text_ru^0.02+text_de^1.5

Russian Query: qf =text_en^0.25+text_ru^1.5+text_de^0.02

English Query: qf =text_en^2+text_ru^0.5+text_de^0.75

Metrics	Previous	Current
ndcg	0.8996	0.8770
bpref	0.7352	0.7305
F0.5	0.1834	0.1833
Map	0.7254	0.7001

The following combination did not give a good result as it ended up fetching many irrelevant documents from other language .

German Query: qf =text_en^0.25+text_ru^0.002+text_de^1.5

Russian Query: qf =text_en^0.25+text_ru^1.5+text_de^0.002

English Query: qf =text_en^2+text_ru^0.5+text_de^0.75

Metrics	Previous	Current
Ndcg	0.8770	0.8288
Bpref	0.7305	0.7239
F0.5	0.1833	0.1834
Map	0.7001	0.6614

There is a slight decrease in all metrics as we have decreased the other language docs boosting, even though it’s higher ranked it came at a later point of time. So as precision increases F precision is slightly increased

German Query : qf=text_en^0.25+text_ru^0.02+text_de^1.5

Russian Query :qf=text_en^0.25+text_ru^1.5+text_de^0.02

English Query :qf=text_en^2+text_ru^1.5+text_de^1.5

Metrics	Previous	Current
ndcg	0.8770	0.8731
bpref	0.7305	0.7311
F0.5	0.1833	0.1834
Map	0.7001	0.6995

Bpref has a slight increase and rest of the metrics have fallen slightly so this is an ideal weightage comibination

8. Query Parser – Hashtag boosting

qf=text_en+text_ru+text_de+tweet_hashtags^0.25

Metrics	Previous	Current
ndcg	0.8770	0.8689
bpref	0.7305	0.7352
F0.5	0.1833	0.1248
Map	0.7001	0.6604

Hashtag boosting resulted in slight increase in bpref as more number of judged documents came early.

9. Query Parser – Text Field+ Hashtag Boosting

German Query : qf=text_en^0.5+text_ru^0.02+text_de^1.5+tweet_hashtags^0.25
Russian Query :qf= qf=text_en^0.5+text_ru^1.5+text_de^0.02+tweet_hashtags^0.25
English Query :qf=text_en^2+text_ru^0.5+text_de^0.5+tweet_hashtags^0.25

Observation : This combination didn't result in very good results as it fetched relevant documents with higher rank later than the relevant with lower ranks because of hashtag weightage for English text field

Metrics	Previous	Current
ndcg	0.8770	0.8730
bpref	0.7305	0.7311
F0.5	0.1833	0.1833
Map	0.7001	0.6995

10. Changing Default parameters for k1 and b

As per previous observations many documents were not fetched in top 10 because of longer lengths, so smoothing 'b' to lower value should fetch more documents

Metric	Previous (with rows 1000)	Current (change of b value to 0.5)
Ndcg	0.8996	0.9015
bpref	0.7352	0.7358
Map	0.7259	0.7294
F0.5	0.1834	0.1834

Observation: There is no change in F0.5 whereas NDCG increased by 0.1, bpref by 0.005 and map by 0.033 .Decreasing value of b resulted in less normalization and fetched more results

11. Decreasing b and increasing value of k

```
<similarity class="solr.BM25SimilarityFactory">
  <float name="k1">1.5</float>
  <float name="b">0.5</float>
</similarity>
```

Metric	Current (change of b value to 0.5)	Current (change of b value to 0.5 and k1=1.5)
ndcg	0.9015	0.9016
bpref	0.7358	0.7332
Map	0.7294	0.7275
F0.5	0.1834	0.1834

Observation: There is no change in F0.5 whereas NDCG increased by 0.001, bpref ,map decreased by minute values . As the result set also had documents with term duplication in the same field smoothening k fetched few more results

12. Decreasing b and decreasing k1

```
<similarity class="solr.BM25SimilarityFactory">
  <float name="k1">1.0</float>
  <float name="b">0.5</float>
</similarity>
```

Metric	Current (change of b value to 0.5)	Current (change of b value to 0.5 and k1=1.0)
ndcg	0.9015	0.9017
bpref	0.7358	0.7358
Map	0.7294	0.7297
F0.5	0.1834	0.1834

Observation: There is no change in F0.5 whereas NDCG increased by 0.002, smoothing parameters b resulted in less normalization and k1 resulted in fetching documents even though they had term repetitions in the same fields

13. Decreasing b to 0.5 and k1 to 1.4

```
<similarity class="solr.BM25SimilarityFactory">
  <float name="k1">1.4</float>
  <float name="b">0.5</float>
</similarity>
```

Metric	Current (change of b value to 0.5)	Current (change of b value to 0.5 and k1=1.4)
ndcg	0.9015	0.9027
bpref	0.7358	0.7358
Map	0.7294	0.7307
F0.5	0.1834	0.1834

Observation: There is no change in F0.5, bpref whereas NDCG increased by 0.012, bpref, map increased by minute value

14. Changing tokenizer to URL tokenizer +qf

solr.UAX29URLEmailTokenizerFactory

As many top k terms were from url(such as http / rt) we tried using url tokenizer , but it wasn't of much help as few documents had keywords in url which were missed now and resulted in low ndcg but bpref improved as it resulted in fetching relevant documents quicker than non relevant

Metrics	Previous	Current
ndcg	0.8996	0.8987
bpref	0.7352	0.7450
F0.5	0.1834	0.1834
Map	0.7259	0.7308

15. Translating Queries

By translating queries to all three languages number of relevant documents retrieved should increase,

As there were not many documents which required translation it didn't fetch great results. It actually ended up fetching higher relevant documents at a later point. It did fetch all relevant documents so precision increased by 0.09

Metrics	Previous	Current
Ndcg	0.8996	0.8863
Bpref	0.7352	0.7042
F0.5	0.1834	0.2329
Map	0.7259	0.6628

16. Translating Queries and Parameters b=0.5 and k1=1.5

Metrics	Previous	Current
ndcg	0.8863	0.8863
bpref	0.7042	0.7029
F0.5	0.2329	0.2329
Map	0.6628	0.6615

17. Translating Queries=0.5 and k1=1.2

Metrics	Previous	Current
ndcg	0.8863	0.8875
bpref	0.7042	0.7029
F0.5	0.2329	0.2329
Map	0.6628	0.6650

18. Translating Queries and Using standard query parser

Standard query parser didn’t turn out to give good results. This may be because there are not many relevant docs for queries in different language other than queries language. Standard query parser wasn’t performing url encoding well like dismax. Special characters such as colon, comma had to be handled separately .Overall dismax performed better

Metrics	Previous	Current
ndcg	0.8863	0.3617
bpref	0.7042	0.4021
F0.5	0.2329	0.2121
map	0.6628	0.3192

19. Hashtag weightage only to German and Russian queries

qf=text_en^1.25+text_ru^0.05+text_de^2+tweet_hashtags^0.25

qf=text_en^1.25+text_ru^2+text_de^0.05+tweet_hashtags^0.25

qf=text_en^2+text_ru^0.5+text_de^0.5

Metrics	Previous	Current
ndcg	0.8863	0.8751
bpref	0.7042	0.7377
F0.5	0.2329	0.1834
Map	0.6628	0.7061

As in the given set there were few English hashtags for complete german doc, query in English should fetch such doc using hashtag. This resulted in increase in 0.03 as more docs which were judged were fetched

20. Using Edismax

Mm=25%

Metrics	Previous	Current
ndcg	0.8863	0.8205
bpref	0.7042	0.6795
F0.5	0.2329	0.2982
Map	0.6628	0.6568

Precision increased as it fetched all relevant docs but the ranking was affected because of mm and rest of the measures had a decrease

21. URL Tokenizer +Dismax + b=0.5 and k1=1.4

Ndcg	0.9041
Bpref	0.7462
Map	0.7387
F.0.5	0.1834

This combination produced better ndcg but the precision is very low as large number of irrelevant documents were retrived. It didn't give bpref as much as ndcg as even though the ranked ordering were optimal , the relevant documents weren't the one to come first than irrelevant

LANGUAGE MODEL

In the language model, 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. Mathematically, this can be represented as $P(q|M_d)$ where M_d is the document model derived from each document.
In Solr, We have two Similarity factories- namely, the LMDirichletSimilarityFactory (which used the Dirichlet smoothing method) and the LMJelineMercerSimilarityFactory (which uses the Jeline Mercer smoothing method). The smoothing parameters for the factories are mu and lambda respectively. Although Jeline Mercer provides better results for longer documents, the Dirichlet smoothing is based on the query size and hence We use the LMDirichletSimilarityFactory as shown below:

Using LMDirichletSimilarityFactory:
The LMDirichletSimilarityFactory is included by adding the following lines to schema.xml :

```
<similarity class="solr.LMDirichletSimilarityFactory">
      </similarity>
```

Before Analysis (at default mu=2000):

ndcg	all	0.8986
set_F.0.5	all	0.1850
map	all	0.6869
bpref	all	0.7029

From the measures, we can see that the F measure is very low because of low precision. Hence we first try tuning the smoothing factor as follows.

Tuning the smoothing factor :
Since the average query size is around 5 we will use mu values which are multiples of 5 (since mu values which are functions of the query size favor performance)

Using mu=100: We get the following trec results

```
<similarity class="solr.LMDirichletSimilarityFactory">
   <str name="mu">100</str>
  </similarity>
```

ndcg	all	0.914
set_F_0.5	all	0.185
map	all	0.7198
bpref	all	0.7192

The F measure shows no difference although the ndcg and map have increased with a slight increase in bpref. This is because even though the precision is not much changed the position in which relevant documents appear have improved.

Using mu=50: We get the following trec results

```
<similarity class="solr.LMDirichletSimilarityFactory">
   <str name="mu">50</str>
  </similarity>
```

ndcg	all	0.9162
set_F_0.5	all	0.185
map	all	0.7302
bpref	all	0.7229

The F measure has not improved but lowering the smoothing factor further will only give redundant results. And also, since the average query size is very small, increasing the smoothing will adversely affect the results as seen by the following trial:

Using mu=4000: We get the following trec results

```
<similarity class="solr.LMDirichletSimilarityFactory">
   <str name="mu">100</str>
  </similarity>
```

ndcg	all	0.8947
set_F_0.5	all	0.185
map	all	0.6804
bpref	all	0.7001

So,we try to optimize the results further by altering the query parameters as follows:

Changing the rows parameter:
We see that using mu=50 give the best results so far. Although, we see that the F0.5 measure for many queries are very low in the range of 0.01 for mu=50. Hence we try to alter the number of irrelevant documents returned by restricting the query rows to 500 as follows:

Query :

```
select?'+en+'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&wt=json&indent=true&debugQuery=true&defType=dismax&qf=text_en+text_de+text_ru&rows=500
```

set_F_0.5	1	0.098
set_F_0.5	2	0.0849
set_F_0.5	3	0.0356
set_F_0.5	4	0.2565
set_F_0.5	5	0.0131
set_F_0.5	6	0.0396
set_F_0.5	7	0.013
set_F_0.5	8	0.0238
set_F_0.5	9	1
set_F_0.5	10	0.0258
set_F_0.5	11	0.75

set_F_0.5	12	0.0355
set_F_0.5	13	0.1935
set_F_0.5	14	0.0922
set_F_0.5	all	0.1901

Before the restriction:

set_F_0.5	1	0.098
set_F_0.5	2	0.0849
set_F_0.5	3	0.0192
set_F_0.5	4	0.2565
set_F_0.5	5	0.0131
set_F_0.5	6	0.0396
set_F_0.5	7	0.013
set_F_0.5	8	0.012
set_F_0.5	9	1
set_F_0.5	10	0.0258
set_F_0.5	11	0.75
set_F_0.5	12	0.0257
set_F_0.5	13	0.1605
set_F_0.5	14	0.0922
set_F_0.5	all	0.185

We can see a slight change in the values for queries 3,8,12, and 13 because the number of irrelevant documents in the result set for these queries has decreased.

When the rows parameter is not set, then the default value for rows which is 10 is used which results in very low values of trec results as seen below:

ndcg	all	0.7942
set_F.0.5	all	0.5716
map	all	0.5967
bpref	all	0.5927

However, the overall F measure is still very low. Next, we try optimizing the query by retrieving documents only in the language that match the language of the query. This when implemented for each query from different languages:

Using Language specific queries:

Analysis on Query 1:

Query:
select?’q=text_en:Russia’s+intervention+in+Syria'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&wt=json&indent=true&debugQuery=true&defType=dismax&rows=500

ndcg	1	0.6757
set_F_0.5	1	0.525
map	1	0.3317

bpref	1	0.345

Here the first query is in English and so only the documents containing text_en:query is retrieved. By doing this, we have achieved a significant increase in F0.5 measure because of increased precision. However, the other measures have lower values because we have reduced the total number of documents retrieved.It can be observed for Russian and German queries also, as seen below.

Analysis on Query 4(German):
Query: select?’q=text_de: Wegen +Flüchtlingskrise:+ Angela+ Merkel+ stürzt+ in+Umfragen'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&wt=json&indent=true&debugQuery=true&defType=dismax&rows=500

ndcg	4	0.7861
set_F_0.5	4	0.6977
map	4	0.4348
bpref	4	0.4348

Analysis on Query 5(Russian):
Query: select?’q= РФ+ в+ Сирии+ вынудили+ 250+ тунисских+ боевиков+ бежать'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&wt=json&indent=true&debugQuery=true&defType=dismax&rows=500

ndcg	all	0.7244
set_F.0.5	all	0.2500
map	all	0.5000
bpref	all	0.5000

From the both the tables above we can see there is only a significant improvement in F measure but otherwise values of all the other measures drop.
The tweet_hashtags field can be considered as a field containing many of the query terms.

Using tweet_hashtags field:
Query:select?’q=text_en:Russia’s+intervention+in+Syria'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&wt=json&indent=true&debugQuery=true&defType=dismax&rows=10000

ndcg	all	0.7962
set_F.0.5	all	0.5793
map	all	0.6019
bpref	all	0.5978

As we can see, this has increased the precision resulting in the increased F measure. But map and bpref values have reduced which indicates that the number of irrelevant documents appearing before relevant has become more.

The next optimization, which we will implement and see, is Query parsing.

Query Parsing:

A Query Parser is a component responsible for parsing the text query and converts it into corresponding Query objects.

Using DisMax Parser:

The DisMax query parser is a disjunction parser across multiple fields with different weights.

The queries executed above used the DisMax query parser. An extended version of the DisMax query parser is the edismax parser.

Using EdisMax Parser:

Using Phrase Fields:

Query:select?'+en+'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&wt=json&indent=true&debugQuery=true&defType=edismax&pf=text_en%5E2+text_de%5E2+text_ru%5E2&qf=text_en+text_de+text_ru'

Here the pf field means "phrase fields" which can be used to "boost" the score of documents in cases where all of the terms in the "q" param appear in close proximity.

ndcg	all	0.8858
set_F.0.5	all	0.2134
map	all	0.7104
bpref	all	0.705

Using the boost 2 for text_en, text_ru,and text_de fields , we have improved the precision. This is because the number of documents with the entire query phrase "as a whole" is boosted to a higher score resulting in higher precision.

Using Minimum Match:

Query:select?'+en+'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&wt=json&indent=true&debugQuery=true&defType=edismax&pf=text_en%5E2+text_de%5E2+text_ru%5E2&qf=text_en+text_de+text_ru&mm=25%25'

The mm field tells us the "minimum "must" match" among the optional clauses in the q field. Specifying mm=25% indicates that at least 25% of the optional clauses must match.

ndcg	all	0.8441
set_F.0.5	all	0.3571
map	all	0.6975
bpref	all	0.709

After including mm=25%, we see an increase in precision (as shown by the F measure) given the restriction that at least 25% of the query clauses must be present for the document to have a higher score.

In order to reduce the effect of one field on the overall result set, we use query boosting. Next we will try query optimization by query boosting as follows:

Query Boosting:

Query boosting involves giving different weights to various fields in the query so as to boost the more relevant documents.

Query:select?'q=text_en:Russia's+intervention+in+Syria'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&wt=json&indent=true&debugQuery=true&defType=dismax&rows=10000&qf=text_en^2+text_de^2+text_ru^2+tweet_hashtags^0.5

Here a boost of 2 is given to all the text fields and the hash tags field is given 0.5 boost which will reduce the effect of the hashtags field on the score of the documents.

ndcg	all	0.7716
set_F.0.5	all	0.5635
map	all	0.5894
bpref	all	0.5849

Here we see that although the precision has increased the bpref value has reduced to very low values, which show the number of judged irrelevant documents has also increased. So we need to compensate this drop in values. For this, we try using the edismax parser instead of dismax as follows:

Query:
select?'+en+'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&wt=json&indent=true&debugQuery=true&defType=edismax&pf=text_en%5E2+text_de%5E2+text_ru%5E2&qf=text_en^2+text_de^2+text_ru^2+tweet_hashtags^0.5&mm=25%25'

Here we have included the phrase field parameter to increase the score of documents that contain the query phrase as a whole and also the minimum match parameter, which will help in reducing the number of irrelevant documents in the result set.

ndcg	all	0.7731
set_F.0.5	all	0.6241
map	all	0.6008
bpref	all	0.6094

Now, we can see a slight increase in map and bpref values since the number of irrelevant documents in the result set is restricted. So we try to use query expansion for further optimization.

Query Expansion:

Query expansion is a process of query reformulation in which additional input about the query is given (in this case, we add additional query terms). A file called "synoyms.txt" is created containing the possible synonyms, which might be used in the place of query terms.
A filter called "SynonymFilterFactory" is added to the fields. At query time, the synonym filter matches the possible synonyms for the query terms as additional query terms and documents containing those terms are scored as well.

Synonyms.txt:

```
intervention =>intermediation, mediation,involution, involvement
Ammo=>arms,weaponry,ammunition,ammo
vulnerable=>dangerous,under attack,endangered,unprotected,unguarded,defenseless
relief=>assuagement, comfort, alleviation…
```

Schema.xml:

```
<analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"
/>
      <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt"
ignoreCase="true" expand="true"/>
      <filter class="solr.LowerCaseFilterFactory"/>
    </analyzer>
```

Query:
select?'+en+'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&wt=json&indent=true&debugQuery=true&defType=edismax&pf=text_en%5E2+text_de%5E2+text_ru%5E2&qf=text_en^2+text_de^2+text_ru^2+tweet_hashtags^0.5&mm=25%25'

ndcg	all	0.7748
set_F.0.5	all	0.5713
map	all	0.5948
bpref	all	0.6054

But as seen from the table, the application of query expansion has only inversely affected the map value because too many irrelevant documents might be retrieved because of the additional query terms.

Next we try to use various filter classes for optimizing the query:

Filters for Query Parsing:

A filter looks at each token in the stream sequentially and decides whether to pass it along, replace it or discard it.

Beider-Morse Filter :

It implements the Beider-Morse Phonetic Matching (BMPM) algorithm, which allows identification of similar names, even if they are spelled differently or in different languages.

Schema.xml:

```
<analyzer>
  <tokenizer class="solr.StandardTokenizerFactory"/>
  <filter class="solr.BeiderMorseFilterFactory" nameType="GENERIC"
ruleType="APPROX” concat="true" languageSet="auto">
  </filter>
</analyzer>
```

Query:

```
select?'+en+'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&wt=json&indent=
true&debugQuery=true&defType=edismax&pf=text_en%5E2+text_de%5E2+text_ru
%5E2&qf=text_en^2+text_de^2+text_ru^2+tweet_hashtags^0.5&mm=25%25'
```

ndcg	all	0.7384
set_F.0.5	all	0.5439
map	all	0.5965
bpref	all	0.6082

Keyword Marker Filter:

When we do not want few keywords to be stemmed, the Keyword Marker Filter is used which takes as input a text file “protwords.txt” that contain a list of protected words. These words will not be stemmed.

Schema.xml:

```
<analyzer>
 <tokenizer class="solr.WhitespaceTokenizerFactory"/>     <filter
class="solr.KeywordMarkerFilterFactory" protected="protwords.txt" />
<filter class="solr.PorterStemFilterFactory" />
</analyzer>
```

Query:

```
select?'+en+'&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&wt=json&indent=
true&debugQuery=true&defType=edismax&pf=text_en%5E2+text_de%5E2+text_ru
%5E2&qf=text_en+text_de+text_ru+tweet_hashtags&mm=25%25’
```

ndcg	all	0.7728
set_F.0.5	all	0.5659
map	all	0.5906
bpref	all	0.6015

Language Translation of Query:

When the given query input is translated to the corresponding languages, a higher boosting is given to the corresponding language fields, which result in the following measures:

ndcg	all	0.906
set_F.0.5	all	0.2331
map	all	0.7153
bpref	all	0.7327

Summary:

The performance of the system is optimized when query is optimized using boosting and language translation in a LMDirichlet Similarity model with a smoothing parameter of mu=50. The best measure for the language model is the ndcg measure because language model as such relies a lot on the frequency of the query terms appearing in the documents.When the normalization factor in ndcg is applied, the measure becomes efficient than other measures, which are based on precision.

VSM Model (Default in Lucene)

The Vector Space Model is the default model being used by Lucene.

We tried tweaking it in various ways like modifying no of max rows in query, changing query parser, hashtag boosting (assigning weightage to and including tweet_hashtags in the queries), query boosting with different sets of trial values, changing tokenizer to URLtokenizer, proximity searches in query parser and language translation. Their corresponding result values have been illustrated in tables given below.

The most efficient tweaking for VSM was when the default Lucene Query Parser was replaced with disMax Query Parser and max rows=1000 was used in the query.

1. Default parameters, default Query Parser, 1000 rows

Metric	**Using Normal Query Parser**
Ndcg	0.1827
Bpref	0.2333
Map	0.0608
F0.5	0.0261

2. Default parameters and disMax Query Parser, 1000 rows

Metrics	**Value for disMax, 1000 rows**
Ndcg	0.9134
Bpref	0.7558
F0.5	0.1834
Map	0.7381
Num_ret	5893
Num_rel	210
Num_rel_retrieved	207

As can be seen from above, disMax Query Parser shows much better results than the default Query Parser present in SOLR.

3. DisMax Query Parser, max rows=10

Metrics	Value for 1000 rows	Value for 10 rows
Ndcg	0.9134	0.8043
Bpref	0.7558	0.6171
F0.5	0.1834	0.5802
Map	0.7381	0.6091

Observation: By decreasing the number of rows fetched from 1000 to 10, an increase was observed in metrics like F0.5 but a decrease was observed in NDCG and BPREF and MAP.

4. DisMax Query Parser and max rows=2000

Metric	Value for 1000 rows	Value for 2000 rows
ndcg	0.9134	0.9111
bpref	0.7558	0.7624
F0.5	0.1834	0.1833
map	0.7881	0.7073

Observation: By increasing the number of rows fetched from 1000 to 2000, metrics like NDCG, F0.5 and MAP slightly decrease and BPREF slightly increased.

5. DisMax Query Parser and HashTag Boosting

Example URL will be as follows:
http://uakk120b34cc.arao4.koding.io:8983/solr/VSM/select?q=Russia%27s+intervention+in+Syria&fl=id%2Cscore%2Ctext_en%2Ctext_ru%2Ctext_de&defType=dismax&qf=text_en+text_ru+text_de+tweet_hashtags&wt=json&indent=true&rows=1000

Metric	Value for 1000 rows	Value with Hashtag
Ndcg	0.9134	0.9111
Bpref	0.7558	0.7624
Map	0.7881	0.7073
F0.5	0.1834	0.1833

Observation: There is slight decrease in NDCG ,MAP and F0.5, very slight increase in BPREF.

6. Query Boosting

As per each query observation if query is in English then it has higher weightage over rest of the languages

If query weightage in German, German doc will have highest weightage, next highest should be English

If query weightage in Russian, Russian doc will have highest weightage, next highest should be English

TRIAL SET 1:

If English:

Text_en^0.8 +text_de^0.1+text_de^0.1

If German:

Text_de^0.8+text_en^0.1+text_ru^0.05

If Russian

Text_ru^0.8+text_en^0.1+text_de^0.05

Metric	Value for 1000 rows	Language based boosting
Ndcg	0.9134	0.9054
Bpref	0.7558	0.7323
Map	0.7381	0.7183
F0.5	0.1834	0.1834

TRIAL SET 2 (with tweet hashtags)

If English :

Text_en^2 +text_de^0.5+text_de^0.5+tweet_hashtags^0.5

If German

Text_de^2+text_en^0.25+text_ru^0.05+tweet_hashtags^0.5

If Russian

Text_ru^2+text_en^0.25+text_de^0.05+tweet_hashtags^0.5

Metric	Value for 1000 rows	Language based boosting
Ndcg	0.9134	0.9045
Bpref	0.7558	0.7293
Map	0.7381	0.7173
F0.5	0.1834	0.1833

TRIAL SET 3

If English :

Text_en^2 +text_de^0.1+text_ru^0.1

If German

Text_de^2+text_en^0.1+text_ru^0.05

If Russian

Text_ru^2+text_en^0.1+text_de^0.05

Metric	Value for 1000 rows	Language based boosting
Ndcg	0.9134	0.9053
Bpref	0.7558	0.7292
Map	0.7381	0.7181
F0.5	0.1834	0.1834

7. Changing tokenizer to URL tokenizer

solr.UAX29URLEmailTokenizerFactory

As many top k terms were from url(such as http / rt) we tried using URL tokenizer

Metric	Value for 1000 rows	URL Tokenizer
Ndcg	0.9134	0.1827
Bpref	0.7558	0.2333
Map	0.7381	0.0608
F0.5	0.1834	0.0261

8. Query Parser – Proximity Searches

Proximity ~5

Metric	Value for 1000 rows	URL Tokenizer
Ndcg	0.9134	0.1827
Bpref	0.7558	0.2333
Map	0.7381	0.0608
F0.5	0.1834	0.0261

9. Translating Queries

By translating queries to all three languages number of relevant documents retrieved should increase

Using disMax query parser and boosting hashtags

Metric	Value for 1000 rows	Using Language Translation
Ndcg	0.9134	0.9077
Bpref	0.7558	0.7150
Map	0.7381	0.6987
F0.5	0.1834	0.2034

Thus, as per all observations made for VSM model, the best way to query is to use disMax query parser and query for max 1000 rows.