

# **Evaluation Of IR Models**

## **Objective:**

The objective of the project is to implement various IR models such as VSM, BM25 and DFR model, evaluate the IR system and improving the search and the MAP value based on our understanding, evaluation and implementation of the IR models.

## **Procedure:**

We are being provided Twitter data in various languages such as English, German and Russian on the topic of European refugee crisis. We performed the following steps to improve the performance in terms of MAP (Mean Average Precision):

- We created different cores implementing different models namely, BM25, VSM and DFR by changing the managed-schema and then indexed the given json file train.json.
- Using the relevance score for each query we evaluated the result using TREC\_eval program.
- Based on the MAP value we got by indexing the original results we made changes by Implementing various techniques such as query booster, synonyms, dismax function and various other techniques.
- Analyzed the results after applying various techniques and then increased the performance by increasing MAP value.

## **IMPLEMENTATION OF IR MODELS:**

### **BM25 Model:**

The term BM stands for Best Matching and this type of model is built on the concepts of probabilistic retrieval. This model uses the ranking function which has the capability of ignoring the interdependence of the words. The concept of tf-idf between the query and document is used by this model.

The model has two parameters :-

**k-** which controls the term frequency distribution

**b-** which controls the tf values through the document normalization, which need to be changed to get better performance in this model.

As in solr this model is implemented by default, so to perform the search we need not make changes in managed schema and solr-config.xml.

### **VSM Model:**

Vector Space Model is an algebraic model in which documents are represented as vectors and the terms as axes. The set of documents represented as vectors are known as Vector State Model. The quantification of document similarity is done using cosine similarity in which calculation of normalized length is done by dot product of document and query. Each document is assigned a score based on the higher dot product or by least angle value.

We implemented the VSM model in Solr by changing the managed-schema as below:

**similarity class:** `<similarity class="org.apache.lucene.search.similarities.ClassicSimilarity"/>`

and changing the solr-config.xml as follows:

```
<libdir="/home/ubuntu/solr/solr-6.2.0/server/solr-webapp/webapp/WEB-INF/lib"    regex="lucene-core-6.2.0.jar" />
```

### **DFR Model:**

Divergence of Randomness is a type of probabilistic model. The weights of the term are computed by measuring the divergence between the term distribution by the random process and the actual distribution in the collection. The main idea of DFR is: The more the divergence of the within-document term-frequency from its frequency within the collection, the more the information carried by the word  $t$  in the document  $d$ .

We implemented the model in Solr by making the changes in the managed schema :

`<similarity class="solr.DFRSimilarityFactory">`

`<str name="c">7.0</str>`

`<str name="normalization">H2</str>`

`<str name="afterEffect">L</str>`

`<str name="basicModel">P</str>`

`</similarity>`

And adding the path to the solr-config.xml :

```
<libdir="/home/ubuntu/solr/solr-6.2.0/server/solr-webapp/webapp/WEB-INF/lib"    regex="lucene-core-6.2.0.jar" />
```

## Various Techniques Applied To Enhance Performance of IR models:

- **Dismax Parser:**

When we search the Twitter data for a given query, if we are searching the data in a particular language we can miss the relevant documents which are in other languages for that respective query. To avoid this loss we use the disjunction function (dismax) which searches the data in all the text fields. For example in the Q001: Russia's intervention in Syria we search the documents in English we lose the relevant documents that is in Russian language and similarly in other queries also.

- **Adding Synonyms:**

Adding synonyms plays a substantial role in increasing the performance of the IR system. So we add the synonyms of the respective words given in the query to the synonyms.txt file. The preference will however be higher for the original terms than the synonyms.

- **Boost Hashtags:**

In a given query if we have hashtags we increase the importance of the hashtags for that query because otherwise Solr treats hashtags as other terms in the query. For Example: in query Q010 #Syria #SALMA #LATAKIA we give more importance to the words in hashtags which results in retrieving of more relevant results and helps in increasing the overall performance.

- **Multilingual Search Query:**

We use this search query to convert the given query into all other languages. Because if we perform search in one specific language then we may lose relevant documents in other languages. So we translate the given query to all other languages and again perform the search on the given data set of Tweets. For example in query Q005 РФ в Сирии вынудили 250 тунисских боевиков бежать we translate this query in English and Russian and we get more number of relevant documents.

- **Changing Value of Parameters:**

In various models we have different kinds of parameters defined for each model. By changing the values of these parameters we can increase the MAP value which will eventually increase the performance.

- **CopyFields:**

Solr provides the copy field by using that copy field by adding field: text\_en, text\_ru and text\_de to \_text\_ helped to increase the performance in various models.

- **Changing the order of the filters:** In the managed- schema of the various models changing the order of the filters of the **field:text\_en** making the filter of **solr.LowerCaseFilterfactory** appear before the **solr.SynonymFilterFactory** and other filters helped to increase the MAP value.

## COMPARISON OF THE MAP VALUES ON VARIOUS IR MODELS USING DIFFERENT TECHNIQUES

### 1. DFR Model:

In DFR Model, we have used various measures like copy fields, dismax parser, using synonyms in the synonyms list, using translation of queries to improve the model. We have also tried changing the various parameters of the similarity class to improve the MAP values. On implementing the hashtags boosting, we did not get the desired MAP values. On comparing the various MAP values from the below chart, the best value for MAP that we got is **0.6964** for the parameters-

**Basic Model- I(F), afterEffect- B, Normalization- H1 and c-7.0**

DFR	
CHANGES MADE	MAP VALUE
Initial	0.6581
Using Copy Fields	0.6710
Using Copy Fields + Dismax Parser	0.6856
Using Copy Fields + Dismax Parser + Synonym+HashTags	0.6882
Using Copy Fields + Dismax Parser + Synonym	0.6889
Using Copy Fields + Dismax Parser + Synonym + Parameter Change in the Similarity Class	0.6964

DFR	
PARAMETER VALUES	MAP VALUE
I(N), B, H2, c=9.0	0.6710
P,L,H2,c=7.0	0.6623
P, B, H1, c=7.0	0.6908
I(F), B, H1, c=7.0	0.6964

The screenshot shows a PuTTY terminal window titled "timberlake.cse.buffalo.edu - PuTTY". The terminal displays a list of performance metrics for a search engine, likely Elasticsearch, comparing an initial state with a state after various enhancements. The metrics include precision at different recall levels (P\_30 to P\_1000), runid, number of queries (num\_q), number of relevant results (num\_rel), number of relevant results returned (num\_rel\_ret), map, gm\_map, Rprec, bpref, recip\_rank, and iprec\_at\_recall at various recall levels (0.00 to 0.80). The values generally show an improvement in performance metrics after the enhancements.

```

P_30      020      0.3000
P_100     020      0.0900
P_200     020      0.0450
P_500     020      0.0180
P_1000    020      0.0090
runid     all      DFR
num_q      all      20
num_ret    all      380
num_rel    all      305
num_rel_ret all      177
map        all      0.6964
gm_map     all      0.6293
Rprec      all      0.6914
bpref      all      0.7015
recip_rank all      1.0000
iprec_at_recall_0.00 all      1.0000
iprec_at_recall_0.10 all      0.9750
iprec_at_recall_0.20 all      0.9464
iprec_at_recall_0.30 all      0.8756
iprec_at_recall_0.40 all      0.8335
iprec_at_recall_0.50 all      0.8026
iprec_at_recall_0.60 all      0.6849
iprec_at_recall_0.70 all      0.5041
iprec_at_recall_0.80 all      0.4245

```

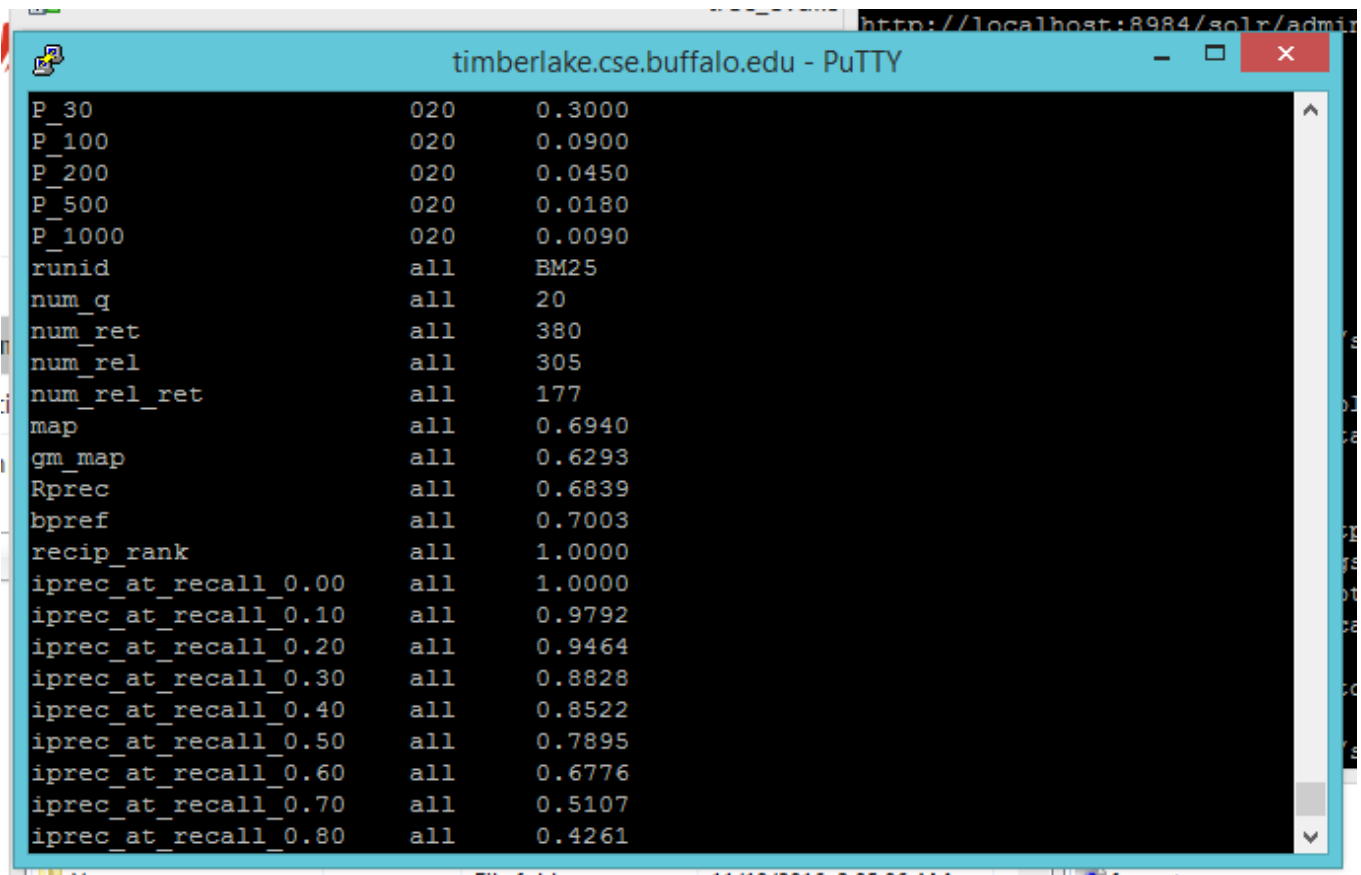
## 2. BM25 Model:

Like DFR model, in BM25 model, we have used various enhancement measures like using copying fields, Dismax Parser, using charFilterClass in the managed-schema to improve the MAP value. From the below tables we can see that the best MAP value that we got for the BM25 model is **0.6940** for the following parameters-

**K1= 0.5, b=0.9**

BM25	
CHANGES MADE	MAP VALUE
Initial	0.6510
Using Copy Fields	0.6623
Using Copy Fields + Dismax Parser	0.6701
Using Copy Fields + Dismax Parser + charFilterClass	0.6708
Using Copy Fields + Dismax Parser + Synonym + Parameter Change in the Similarity Class(k1=0.5, b=0.9)	0.6940

BM25		
K1	B	Map
-	-	0.6623
1.35	0.6	0.6617
1.2	0.6	0.6841
0.5	0.9	0.6940



The screenshot shows a PuTTY terminal window titled "timberlake.cse.buffalo.edu - PuTTY". The terminal displays a list of configuration parameters and their corresponding values, likely from a search engine configuration file. The parameters include P\_30, P\_100, P\_200, P\_500, P\_1000, runid, num\_q, num\_ret, num\_rel, num\_rel\_ret, map, gm\_map, Rprec, bpref, recip\_rank, and various iprec\_at\_recall values. The values for P\_30 through P\_1000 are 0.3000, 0.0900, 0.0450, 0.0180, and 0.0090 respectively. The runid is set to all, and the BM25 algorithm is selected. The map value is 0.6940, which matches the value in the BM25 table above. The bpref value is 0.7003, which is the value mentioned in the text as the final MAP value after improvement.

P_30	020	0.3000
P_100	020	0.0900
P_200	020	0.0450
P_500	020	0.0180
P_1000	020	0.0090
runid	all	BM25
num_q	all	20
num_ret	all	380
num_rel	all	305
num_rel_ret	all	177
map	all	0.6940
gm_map	all	0.6293
Rprec	all	0.6839
bpref	all	0.7003
recip_rank	all	1.0000
iprec_at_recall_0.00	all	1.0000
iprec_at_recall_0.10	all	0.9792
iprec_at_recall_0.20	all	0.9464
iprec_at_recall_0.30	all	0.8828
iprec_at_recall_0.40	all	0.8522
iprec_at_recall_0.50	all	0.7895
iprec_at_recall_0.60	all	0.6776
iprec_at_recall_0.70	all	0.5107
iprec_at_recall_0.80	all	0.4261

### 3. VSM Model

In VSM model, we have again used copy fields, dismax parser, and added synonyms in the the synonyms.txt file to improve the MAP value. The final MAP value that we got after improvement is **0.7010**.

VSM	
CHANGES MADE	MAP VALUE

Initial	0.6418
Using Copy Fields	0.6532
Using Copy Fields + Dismax Parser	0.6915
Using Copy Fields + Dismax Parser + Synonym	0.7010

Screenshot:

P_1000	020	0.0090
runid	all	VSM
num_q	all	20
num_ret	all	380
num_rel	all	305
num_rel_ret	all	176
map	all	0.7010
gm_map	all	0.6384
Rprec	all	0.6831
bpref	all	0.7161
recip_rank	all	1.0000
iprec_at_recall_0.00	all	1.0000
iprec_at_recall_0.10	all	0.9938
iprec_at_recall_0.20	all	0.9464
iprec_at_recall_0.30	all	0.8828
iprec_at_recall_0.40	all	0.8531
iprec_at_recall_0.50	all	0.7824
iprec_at_recall_0.60	all	0.6611
iprec_at_recall_0.70	all	0.5472
iprec_at_recall_0.80	all	0.4361
iprec_at_recall_0.90	all	0.3153
iprec_at_recall_1.00	all	0.3153
P_5	all	0.9000
P_10	all	0.6850
P_15	all	0.5267
P_20	all	0.4400
P_30	all	0.2933
P_100	all	0.0880
P_200	all	0.0440
P_500	all	0.0176
P_1000	all	0.0088