IR Project 3

BY

DARSHAN MHATRE 50208274 (dmhatre)
DIVYANSH BHATNAGAR 50206553 (dbhatnag)

IR Project 3

OBJECTIVE:

The goal of this project is to implement various IR models, evaluate the IR system and improve the search result based on the implementation and evaluation of different models.
We implement three IR models, namely Vector Space Model (VSM), Best Matching (BM25) and Divergence From Randomness (DFR) model.

TOOLS USED:

Solr-6.2.1 : Used to index given data and implement the various IR models.

TREC_eval program: Used to evaluate the result sets from the implemented IR models.

IR Models:

- **Vector Space Model (VSM):**
  The representation of a set of documents as vectors in a common vector space is known as the vector space model and is fundamental to a host of information retrieval operations ranging from scoring documents on a query, document classification and document clustering.

- **Best Matching Model (BM25):**
  BM25 is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of the inter-relationship between the query terms within a document (e.g., their relative proximity). It is not a single function, but actually a whole family of scoring functions, with slightly different components and parameters.

  The BM25 implementation depends on two parameters:
  k1: controls the term frequency normalization
  b: controls the tf values through document normalization.

- **Divergence from Randomness Model (DFR):**
The Divergence from Randomness (DFR) paradigm is based on, Harter's 2-Poisson indexing-model. Term weights are computed by measuring the divergence between a term distribution produced by a random process and the actual term distribution.

PROCEDURE:

We are given twitter data and relevance score for all the documents. We follow the given steps to implement the given IR models:

1. Modification of the default schema according to each model:
   A. **BM25**: We use the BM25SimilarityFactory to implement the BM25 model.
   B. **VSM:** We use ClassicSimilarityFactory to implement the VSM model.
   C. **DFR:** We use DFRSimilarityFactoryt to implement the DFR model.

2. We index the data according to the above mentioned implementation.

3. We calculate the score for each document and use the trec_eval program to calculate the mean average precision (MAP) to use it as reference.

4. We use various techniques to improve the performance for each model based on the MAP value.

TECHNIQUES:

The various techniques which were implemented to improve the performance of the models are:

1. **Using dismax query parser:**
The default query parser in solr is known as the lucene parser. The drawback of this parser is that it is very intolerant to syntax errors. Therefore, we use the Disjunction Max Query Parser (DisMax parser). This parser is designed to process simple phrases entered by user and search different terms across several fields using different weighting based on the significance of each field.

2. **Boosting fields containing query terms and boost query:**
For any given query, there are certain terms which are more important/relevant than others. In our DisMax parser we define the parameters qf, pf and bq to improve the relevance of the document.
Query Fields(qf) specifies the fields in the index on which we perform the query. For example, qf = text_de^2 + text_ru^2 + text_en^2 will boost these mentioned fields for a query match.

Phrase fields(pf) boosts the score of document in cases where all terms in the query appear in close proximity. For example, pf = text_de^2 + text_ru^2 + text_en^2
Boost Query(bq) specifies a factor by which a term or phrase should be boosted in importance when considering a match. For example, bq = text_ru:Russia^2 + Syria^2 will boost the tem Russia and Syria in the text_ru field.

3. **Adding Synonyms:**
Synonyms play a crucial role in improving the performance of an IR model. For a given query the user might type a different term but has essentially the same meaning as a standard query. Addition of synonyms enhances the scope of query parsing and retrieves more logically relevant results. We should be careful while implementing synonyms as we want to retrieve the documents for terms in the query and then the documents for the synonyms of the query. For example, terms like Refugee, terrorist, civil war, crisis we added synonyms in all our available language i.e. Russian, German and English, Refugee, Беженцев, Flüchtling and terrorist, террорист, Terroristen.

4. **Boosting the hashtags field for queries containing hashtags:**
When we get a query containing a hashtag, the document containing that hashtag is given more importance over other documents. We use the Boost Query function of the DisMax Parser to implement this. For example, bq=tweet_hashtags:Salma^2 + Syria^2, this will boost the documents with hashtags containing Russia and Syria.

5. **Using multilingual search query to query in all languages:**
In certain scenarios when we try to hit a query in one language we might not get as many relevant documents as we will get if the query is converted in other languages. To implement it we converted each of the given query in all available languages i.e. Russian, German and English, and then retrieved the documents.

6. **Using CopyField**:
Solr provides a functionality of using CopyField to copy the value of one field into another. We copy the contents of text_en, text_de and text_ru into a general field _text_, this impacts the performance of the IR system in such a way that we query only the _text_ field directly instead of querying individual fields.
For example,
```
<copyField source="text_de" dest="_text_"/>
<copyField source="text_en" dest="_text_"/>
<copyField source="text_ru" dest="_text_"/>
```

7. **Using Filters:**
In an attempt to increase the number of relevant documents, we wanted to avoid terms in the urls present in tweets(mainly to avoid retweets). Along with that we made use of some other relevant filters which we learned in our first IR project.
For Example,

```
 <charFilter class="solr.PatternReplaceCharFilterFactory" pattern="#\S+" replacement=""/>
     <charFilter class="solr.PatternReplaceCharFilterFactory" pattern="@" replacement=""/>
     <charFilter class="solr.PatternReplaceCharFilterFactory" pattern="-" replacement=""/>
     <charFilter class="solr.PatternReplaceCharFilterFactory"
pattern="((http[s]?|ftp):\/)?\/?([^:\/\s]+)((\/\w+)*\/)([\w\-\.]+[^#?\s]+)(.*)?(#[\w\-]+)?"
replacement=""/>
```

8. **Playing around with parameters:**
The BM25 and DFR IR models have different parameters which impact the performance of the given model. The DFR model has BasicModels, AfterEffect and Normalization parameters, and the BM25 has k1 and b parameters, which are specified in the documentation. After going through the documentation thoroughly we discovered the parameters which impacted the performance the most. {}

PERFORMANCE ENHANCEMENT:

- **Vector Space Model:**

In order to improve the performance, the very first thing which we did in this and all other models was to use filter techniques learned in our First IR project.

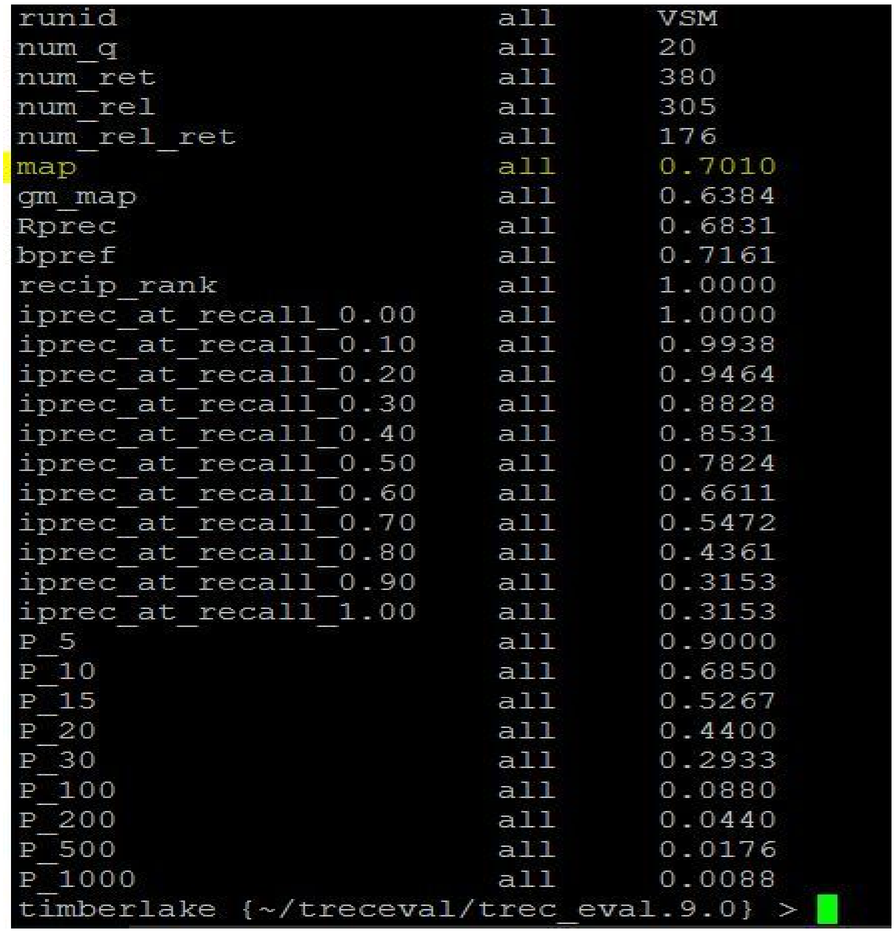
Then in VSM model, we tried all possible combinations of all the techniques mentioned above except technique 8 which involved playing around with parameters(Since, ClassicSimilarityFactory has no parameters.).

First breakthrough we got was through the use of DisMax Parser. Then we tried various permutations of field boosts which improved our MAP values significantly.

Then we made use of CopyFields which initially lowered our MAP values. But, then we adjusted our field boosts accordingly and eventually ended up improving MAP values to a satisfactory level.

In the end, we made use of Synonyms which slightly lifted up our MAP value.

Techniques Used	Initial	DisMax Parser with Field Boosting	DisMax Parser + CopyField	DisMax Parser + CopyField + Synonyms
MAP Values	0.6420	0.6856	0.6978	0.7010



```
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
timberlake {~/treceval/trec_eval.9.0} >
```

FINAL MAP VALUE

- **BM25 model:**

  After basic use of filter techniques, most of the performance improvement of BM25 model was done by playing around with the parameters present in the BM25SimilarityFactory. We spent quality amount of time in understanding these parameters and how & why these parameters affect the performance of the IR system.

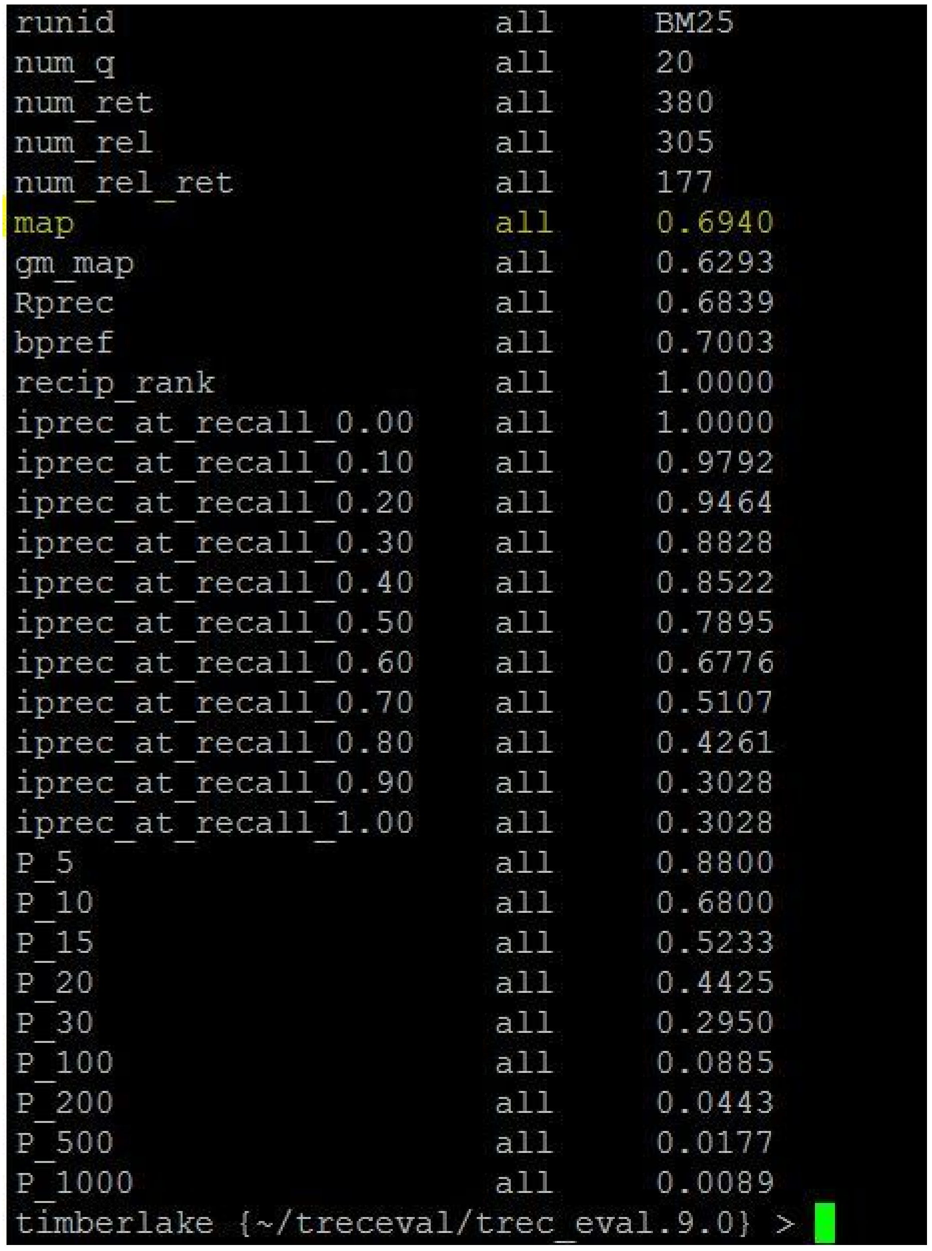
  Then, we moved on to other techniques mentioned above. Out of those, dismax query parser gave significant improvement in the MAP value when used with field boosts

  Then again we introduced the CopyField in a similar way as we did in the improvement of VSM model.

Techniques Used	Initial	Parameter Tweaking	Parameter Tweaking + DisMax parser	Parameter Tweaking + DisMax parser + CopyField
MAP Values	0.6581	0.6784	0.6901	0.6940

Though, the combinations of k1 and b values we tried was very large, we have mentioned few set of values in below mentioned table which proved to be reference values in our attempt to find a increasing/decreasing trend in the MAP values w.r.t. Variations in (k1,b).

k1	b	MAP
1.2	0.75	0.6581
1.4	0.8	0.6510
1.2	0.5	0.6430
0.8	0.6	0.6690
0.5	0.9	0.6784



```
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
iprec_at_recall_0.90    all     0.3028
iprec_at_recall_1.00    all     0.3028
P_5                     all     0.8800
P_10                    all     0.6800
P_15                    all     0.5233
P_20                    all     0.4425
P_30                    all     0.2950
P_100                   all     0.0885
P_200                   all     0.0443
P_500                   all     0.0177
P_1000                  all     0.0089
timberlake {~/treceval/trec_eval.9.0} >
```

FINAL MAP VALUE

- **Divergence from Randomness:**

  After going through various documentations provided for the project, we reached to the conclusion that parameter tweaking is the key to improve performance of DFR model.

  After going through the concepts of various parameters present in DFRSimilarityFactory we established that the following parameters impacted the performance the most:

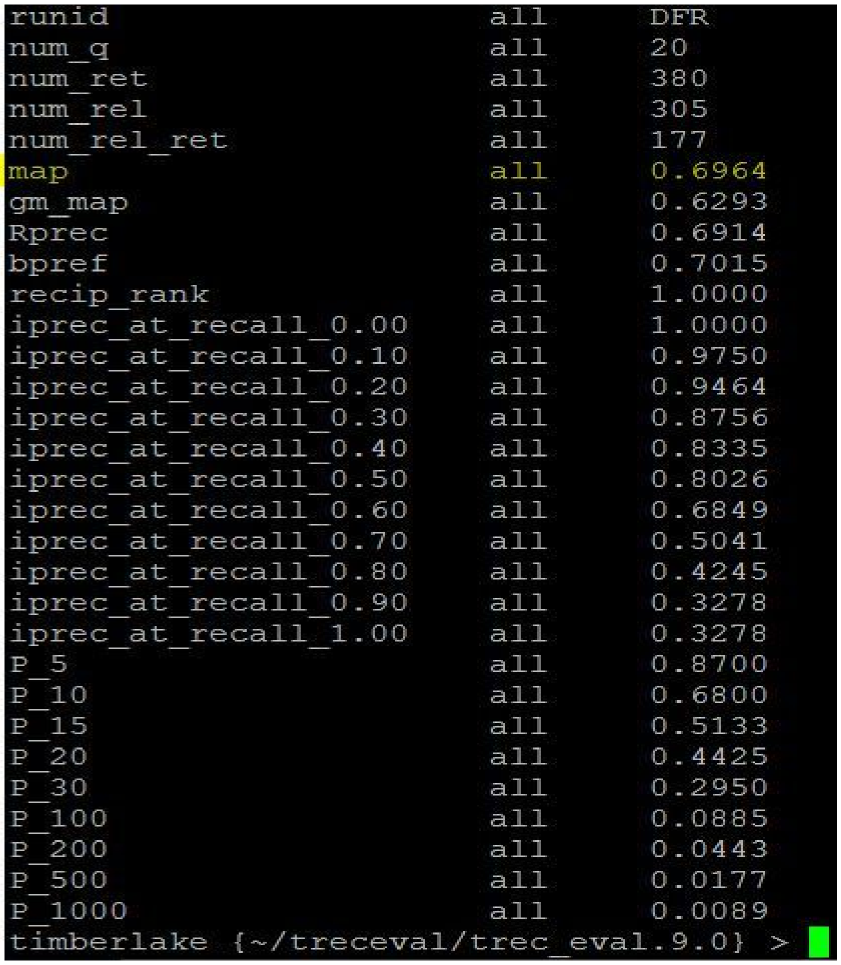
BasicModelG: Geometric approximation of Bose-Einstein
BasicModelP: Poisson approximation of the Binomial
BasicModelD: Divergence approximation of the Binomial
BasicModelIF: Inverse term frequency [approximation of I(ne)]
AfterEffectL: Laplace's law of succession
AfterEffectB: Ratio of two Bernoulli processes
NormalizationH1: Uniform distribution of term frequency
NormalizationH2: term frequency density inversely related to length
NormalizationH3: term frequency normalization provided by Dirichlet prior
NormalizationZ: term frequency normalization provided by a Zipfian relation

Some of the key values of parameters which helped us in understanding the trend in changes in the MAP values are mentioned below.

BasicModel	AfterEffect	Normalization	MAP Value
G	B	H2	0.6591
P	B	H1	0.6674
IF	B	H1	0.6719

Then we tried all other techniques as we did in rest of the two models and improved our system to a great extent.

Techniques Used	Initial	Parameter Tweaking	Parameter Tweaking + DisMax parser	Parameter Tweaking + DisMax parser + CopyField
MAP Values	0.6591	0.6719	0.6915	0.6964



```
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
iprec_at_recall_0.90          all      0.3278
iprec_at_recall_1.00          all      0.3278
P_5                           all      0.8700
P_10                          all      0.6800
P_15                          all      0.5133
P_20                          all      0.4425
P_30                          all      0.2950
P_100                         all      0.0885
P_200                         all      0.0443
P_500                         all      0.0177
P_1000                        all      0.0089
timberlake {~/treceval/trec_eval.9.0} >
```

FINAL MAP VALUE

INFERENCE:

In this project, First we learned the basic implementation of three different IR system models i.e. VSM, BM25,DFR.

Then with the help of trec_eval software we checked and compared MAP values after making various changes in the system. By the end of this project we managed to improve the performance of all the three models by a significant amount.

For generalized system, VSM model proved to be most efficient model after all the efforts made for improvement.

References

1) (http://archive.apache.org/dist/lucene/solr/ref-guide/apache-solr-ref-guide-6.2.pdf)
2) http://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html?is-external=true
3) http://trec.nist.gov/trec_eval/
4) https://cwiki.apache.org/confluence/display/solr/The+Standard+Query+Parser