CSE 535: INFORMATION RETRIEVAL
PROJECT 3

EVALUATION OF IR MODELS

NAME: Akshara Santharam

UBIT NAME: aksharas

PERSON NUMBER: 50313950

OVERVIEW

The goal of the project is to implement various IR models, evaluate the IR system and improve the search results based on the understanding, implementation and evaluation of the models. Given twitter data in three languages – English, German and Russian, 15 sample queries and the corresponding relevance judgements, twitter data must be indexed using Solr and the following three IR models: (i) Language Model (ii) BM25 and (iii) Divergence from Randomness (DFR) Model. The results from these three sets will be evaluated using Trec_eval program. Based of the evaluation results, an attempt is made to improve the performance in terms of Mean Average Precision (MAP).

DATASET

The data to be used is Twitter data in json format, training_tweet.json. Three languages are included- English(text_en), German(text_de) and Russian(text_ru). The training_tweet.json file contains approximately 3500 tweets with some fields extracted from raw data. The sample tweet format is as follows:

```
{

"lang": ,
"id": ,
"text_de": ,
"text_en": ,
"text_ru": ,
"tweet_urls": [],
"tweet_hashtags": []

}
```

IMPLEMENTING THE DEFAULT CONFIGURATIONS OF THE IR MODELS

1. LANGUAGE MODEL

The Language Model can be implemented as a global configuration using the following similarity class in the schema.xml file:

```
<similarity class=”solr.LMDirichletSimilarityFactory”>
        <float name=”mu”> 2000 </float>
</similarity>
```

After indexing the training_tweet.json provided for the configured schema.xml for the core on solr, TREC_eval is run to evaluate the sample query output file.

./trec_eval -q -c -M 1000 qrel.txt sample_query_output.txt

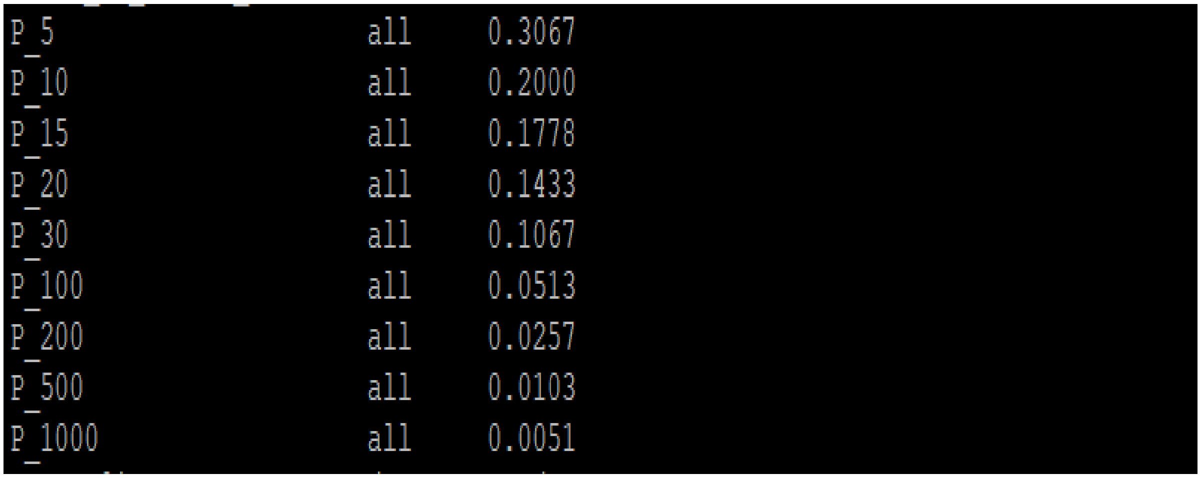
The above command will give the number of common evaluation measure results. The screenshot for the above is as shown below:

```
ubuntu@ip-172-31-22-27:~/solr-8.2.0/trec_eval-9.0.7$ ./trec_eval -q -c -M 1000 q
rel.txt sample_query_output.txt
num_ret                 001     289
num_rel                 001     20
num_rel_ret             001     20
map                     001     0.5707
Rprec                   001     0.5500
bpref                   001     0.4950
recip_rank              001     1.0000
iprec_at_recall_0.00    001     1.0000
iprec_at_recall_0.10    001     1.0000
iprec_at_recall_0.20    001     1.0000
iprec_at_recall_0.30    001     0.8571
iprec_at_recall_0.40    001     0.6111
iprec_at_recall_0.50    001     0.6111
iprec_at_recall_0.60    001     0.5000
iprec_at_recall_0.70    001     0.2632
iprec_at_recall_0.80    001     0.2609
iprec_at_recall_0.90    001     0.2609
iprec_at_recall_1.00    001     0.2564
P_5                     001     1.0000
P_10                    001     0.6000
P_15                    001     0.5333
P_20                    001     0.5500
P_30                    001     0.4333
P_100                   001     0.2000
P_200                   001     0.1000
P_500                   001     0.0400
P_1000                  001     0.0200
num_ret                 002     325
num_rel                 002     19
num_rel_ret             002     19
map                     002     0.7629
Rprec                   002     0.6316
bpref                   002     0.7072
recip_rank              002     1.0000
iprec_at_recall_0.00    002     1.0000
iprec_at_recall_0.10    002     1.0000
```

Figure 1

```
iprec_at_recall_0.20    002     1.0000
iprec_at_recall_0.30    002     1.0000
iprec_at_recall_0.40    002     0.8333
iprec_at_recall_0.50    002     0.8333
iprec_at_recall_0.60    002     0.8000
iprec_at_recall_0.70    002     0.5769
iprec_at_recall_0.80    002     0.5000
iprec_at_recall_0.90    002     0.4390
iprec_at_recall_1.00    002     0.3800
P_5                     002     1.0000
P_10                    002     0.8000
P_15                    002     0.8000
P_20                    002     0.6000
P_30                    002     0.5000
P_100                   002     0.1900
P_200                   002     0.0950
P_500                   002     0.0380
P_1000                  002     0.0190
num_ret                 003     873
num_rel                 003     12
num_rel_ret             003     12
map                     003     0.8040
Rprec                   003     0.7500
bpref                   003     0.7167
recip_rank              003     1.0000
iprec_at_recall_0.00    003     1.0000
iprec_at_recall_0.10    003     1.0000
iprec_at_recall_0.20    003     1.0000
iprec_at_recall_0.30    003     1.0000
iprec_at_recall_0.40    003     1.0000
iprec_at_recall_0.50    003     1.0000
iprec_at_recall_0.60    003     0.8182
iprec_at_recall_0.70    003     0.8182
iprec_at_recall_0.80    003     0.3636
iprec_at_recall_0.90    003     0.3636
iprec_at_recall_1.00    003     0.3636
```

Figure 2



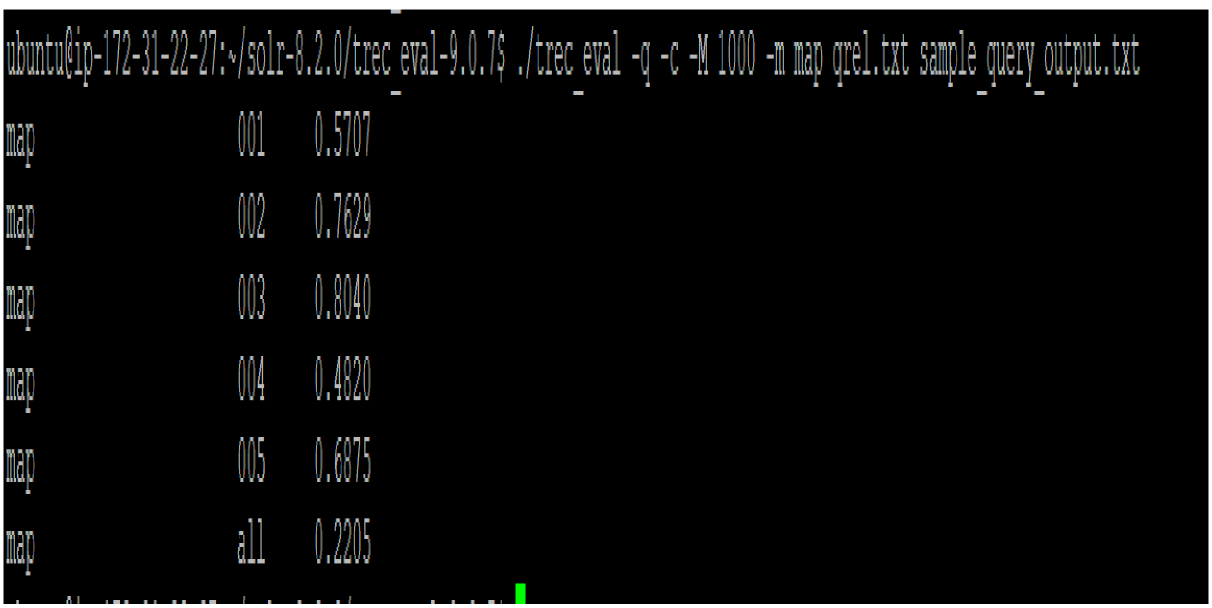
```
P_5                    all     0.3067
P_10                   all     0.2000
P_15                   all     0.1778
P_20                   all     0.1433
P_30                   all     0.1067
P_100                  all     0.0513
P_200                  all     0.0257
P_500                  all     0.0103
P_1000                 all     0.0051
```

Figure 3

-m option can be used to specify the measure we prefer. This command will give the map measure result for each query followed by overall performance.

./trec_eval -q -c -M 1000 -m map qrel.txt sample_query_output.txt

The screenshot for the above is as shown below:



```
ubuntu@ip-172-31-22-27:~/solr-8.2.0/trec_eval-9.0.7$ ./trec_eval -q -c -M 1000 -m map qrel.txt sample_query_output.txt
map                    001     0.5707
map                    002     0.7629
map                    003     0.8040
map                    004     0.4820
map                    005     0.6875
map                    all     0.2205
```

Figure 4

2. BM25

We can implement BM25 model using the following Similarity class in the schema.xml

<similarity class=”solr.BM25SimilarityFactory”>
<str name=”k1”>1.2</str>
<str name=”b”>0.7</str>
</similarity>

After indexing the training_tweet.json provided for the configured schema.xml for the core on solr, TREC_eval is run to evaluate the sample query output file.

./trec_eval -q -c -M 1000 qrel.txt sample_query_output.txt

The above command will give the number of common evaluation measure results.

-m option can be used to specify the measure we prefer. This command will give the map measure result for each query followed by overall performance.

./trec_eval -q -c -M 1000 -m map qrel.txt sample_query_output.txt

The screenshot for the above is as shown below:

```
map                   001     0.3418
map                   002     0.3913
map                   003     0.5729
map                   004     0.6130
map                   005     0.5000
map                   006     0.4926
map                   007     0.8333
map                   008     1.0000
map                   009     1.0000
map                   010     1.0000
map                   011     1.0000
map                   012     0.6586
map                   013     0.1022
map                   014     0.5577
map                   015     0.8667
map                   016     0.9107
map                   017     0.2857
map                   018     0.6110
map                   019     1.0000
map                   020     0.4118
map                   all     0.6575
gm_map                all     0.5829
```

Figure 5

3. DIVERGENCE FROM RANDOMNESS (DFR)

We can implement DFR model using the following Similarity class in the schema.xml

```
<similarity class=”solr.DFRSimilarityFactory”>
        <str name=”basicModel”>G</str>
        <str name=”afterEffect”>B</str>
        <str name=”normalization”>H2</str>
</similarity>
```

After indexing the training_tweet.json provided for the configured schema.xml for the core on solr, TREC_eval is run to evaluate the sample query output file.

./trec_eval -q -c -M 1000 qrel.txt sample_query_output.txt

The above command will give the number of common evaluation measure results.

-m option can be used to specify the measure we prefer. This command will give the map measure result for each query followed by overall performance.

./trec_eval -q -c -M 1000 -m map qrel.txt sample_query_output.txt

The screenshot for the above is as shown below:

```
map                     001     0.3735
map                     002     0.3923
map                     003     0.5471
map                     004     0.6130
map                     005     0.5000
map                     006     0.4991
map                     007     0.8333
map                     008     1.0000
map                     009     1.0000
map                     010     1.0000
map                     011     0.9861
map                     012     0.7495
map                     013     0.1041
map                     014     0.5942
map                     015     0.8667
map                     016     0.8626
map                     017     0.2857
map                     018     0.6168
map                     019     1.0000
map                     020     0.4118
map                     all     0.6618
gm_map                  all     0.5891
```

Figure 6

OPTIMIZING THE MODELS

For optimizing the models, use Mean Average Precision (MAP). Mean average precision for a set of queries is the mean of the average precision scores for each query.

$$\mathrm{MAP} = \frac{\sum_{q=1}^{Q} \mathrm{AveP(q)}}{Q}$$

Where Q is the number of queries.