PROJECT 3: EVALUATION OF IR MODELS

NIKHIL P. YADAV UBIT: nyadav2

Overview

The goal of this project is to implement various IR models, evaluate the IR system and improve the search results based on our understanding of the models, the implementation and the evaluation.
The three IR models to be implemented are

1. **Best matching 25 (BM25) model**
2. **Divergence From Randomness (DFR) model**
3. **Language Model**

Input data is twitter data in three languages: English, German and Russian which is to be indexed using *Solr* and the results are evaluated using the *trec_eval* program. Based on these results we have to improve the Mean Average Precision (MAP) results.

Model Implementation

Initial steps common for all the models are as follows:

i. Create directories using FileZilla with the core names (BM25, DFR, LM) inside the server.(Note: FileZilla is used to create a connection between local system and the server).
ii. Create *conf* and *data* directories in these cores.
iii. Start Solr. A default core *gettingstarted* is created. Stop Solr.
iv. Copy the *conf* data from getting started to all the three cores. Start Solr.
v. Post the *twitter_training.json* file on all the cores.
vi. Copy the *managed_schema* file on to the local system and rename it to *schema.xml.* Stop Solr.
vii. Add similarity classes as per the model to the *schema.xml* and copy it to the cores.(Note: **delete** the previously created *managed_schema* file and *schema.xml.bak* before posting the data again).
viii. Post the *twitter_training.json* again using the new schema that has a similarity class as per the model.
ix. Now, run the *json_to_trec.py* script to get the output in *trec_eval* format.
x. Feed the generated output to *trec_eval* executable to get MAP scores for the queries.
xi. Tweak the hyper-parameters in the similarity class to improve these MAP scores for each model.
xii. After selecting optimal values for hyperparameters post the *test_queries.txt* file to generate the *trec_eval* formatted output which will be fed to the *trec_eval* executable later for relevance judgment.

- **BM25 Model**

  BM25 is a bag of words retrieval function which is used by search engines to rank the documents according to relevance of the documents to the queries. The simplest scoring for a document in this model is the IDF weighting for the query terms.
  For this model the default values of the hyper-parameters are *k1 = 1.2* and *b = 0.75*
  The steps to implement this model are:

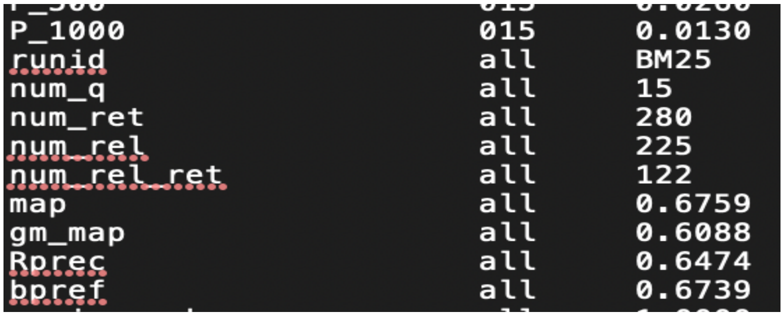
  1. As per the initial steps, the similarity class is as follows,

```
 <similarity class="solr.BM25SimilarityFactory">
   <str name="b">0.9</str>
  <str name="k1">1.2</str>
</similarity>
```

2. In the above similarity class '*b*' and '*k1*' are the two hyper-parameters that are tuned in order to get a better MAP score.
3. The different values used for this model are present in table below

k1	***b***	***MAP***
1.2	0.75	0.6756
1.2	*0.9*	*0.6759*
1.4	0.9	0.6744
2.0	0.9	0.6724

4. As seen from the above table the second row gives a optimum value amongst the others.
5. From this we can see that the values with increasing value of '*k*' decrease while values with increasing values of *'b'* increase.
6. Therefore, increasing the values of *'b'* and decreasing the values of *'k'* will yield a better MAP score.
7. *trec_eval* screenshot for this model,



```
P_1000                  015     0.0130
runid                   all     BM25
num_q                   all     15
num_ret                 all     280
num_rel                 all     225
num_rel_ret             all     122
map                     all     0.6759
gm_map                  all     0.6088
Rprec                   all     0.6474
bpref                   all     0.6739
```

- **DFR Model**

  Divergence From Randomness model is a probabilistic model. For this specific model we use "BasicModelG" plus "Bernoulli" first normalization plus "H2" second normalization.
    - The *'BasicModelG'* is geometric approximation of Bose-Einstein
    - The *'AfterEffectB'* is ratio of two Bernoulli processes
    - The *'H2'* is term-frequency density inversely related to length
    - Finally *'c'* is the hyper-parameter used to tune the model that controls term frequency normalization w.r.t. document length, where default value of *c = 1.*

  The steps for this model are as follows,

1. As per the initial steps the similarity model is as follows,

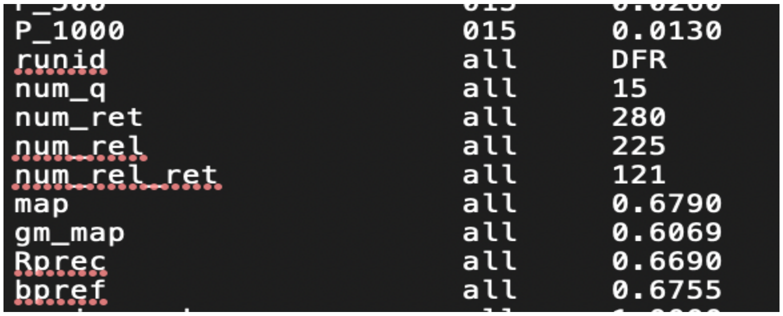
```
<similarity class="solr.DFRSimilarityFactory">
  <str name="basicModel">G</str>
  <str name="afterEffect">B</str>
  <str name="normalization">H2</str>
  <float name="c">15</float>
</similarity>
```

2. In the above similarity class *'c'* is the hyper-parameter that needs to be tuned in order to get a better MAP score.
3. The different values of *'c'* are as shown in the table below,

c	MAP
1	0.6750

7	0.6760
10	0.6760
15	*0.6790*

4. As seen from the above table the last row gives a optimum value amongst the others.
5. From the above observations we can conclude that with increasing *‘c’* value the MAP score increases up to a certain limit.
6. *trec_eval* screenshot for this model,



```
P_1000                 015      0.0130
runid                  all      DFR
num_q                  all      15
num_ret                all      280
num_rel                all      225
num_rel_ret            all      121
map                    all      0.6790
gm_map                 all      0.6069
Rprec                  all      0.6690
bpref                  all      0.6755
```

- **LM**

Language model is a probability distribution over sequence of words. LM has only one hyper-parameter that can be tuned to get a better MAP score called *‘mu’.* It is a smoothing parameter with a default value *mu = 2000.*

The steps for this model are as follows,

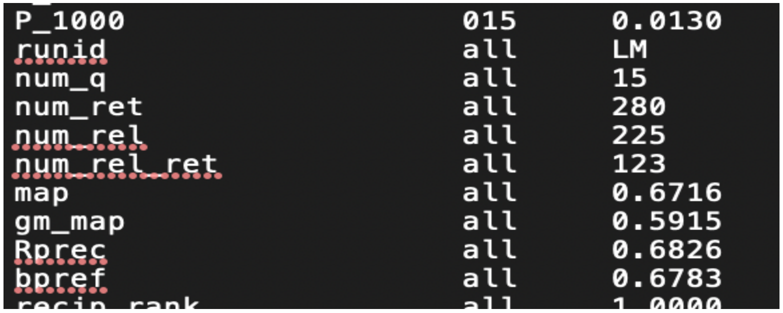
1. As per the initial steps the similarity model is as follows,

```
<similarity class="solr.LMDirichletSimilarityFactory">
   <float name="mu">300</float>
  </similarity>
```

2. The different values of *‘mu’* are as shown in the table below,

mu	MAP
2000	0.6135
1000	0.6297
500	0.6632
300	*0.6716*

3. As seen from the above table the last row gives a optimum value amongst the others.
4. From the above observations we can conclude that with decreasing *‘mu’* the MAP score increases.
5. *trec_eval* screenshot for this model,



```
P_1000                 015      0.0130
runid                  all      LM
num_q                  all      15
num_ret                all      280
num_rel                all      225
num_rel_ret            all      123
map                    all      0.6716
gm_map                 all      0.5915
Rprec                  all      0.6826
bpref                  all      0.6783
recip_rank             all      1.0000
```