EVALUATION OF IR MODELS

Project 3

Haril Anil Satra	*harilani*
Arun Sharma	*arunshar*

UNIVERSITY AT BUFFALO
CSE 535: Information Retrieval

Goal:

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

Implementation of the three models:

Vector Space Model:

Adding the following line in schema.xml implements the Vector Space Model (VSM)

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

MAP Value for the training queries after implementing VSM: 0.6418

BM25 Model:

Adding the following lines in schema.xml implements the BM25 Model

The default value of parameters is k=1.2 and b=0.75, however these can be changed according to the need.

```
<similarity class="solr.BM25SimilarityFactory">
  <str name="k1">1.2</str>
  <str name="b">0.75</str>
</similarity>
```

MAP Value for the training queries after implementing BM25 using above default parameters: 0.6575

Divergence From Randomness Model:

Adding the following lines in schema.xml implements the DFR Model

```
<similarity class="solr.DFRSimilarityFactory">
  <str name="basicModel">G</str>
  <str name="afterEffect">B</str>
  <str name="normalization">H2</str>
  <float name="c">7</float>
</similarity>
```

MAP Value for the training queries after implementing DFR using above default parameters: 0.6588

Techniques to improve the MAP score:

1. Tweaking the parameters of the BM25 Model

   k1 (float): Controls non-linear term frequency normalization (saturation).
   The variable k1 is a positive tuning parameter that calibrates the document-term frequency scaling.
   b (float): Controls to what degree document length normalizes tf values.
   The variable b lies between 0 and 1 inclusive and determines the scaling by document length.

   Experimenting with different values of the parameters of BM25 Model. Some of the combinations are as follows:

PARAMETERS(k1,b)	**MAP VALUE**
1.2,0.75	0.6818
1.4,0.6	0.6809
1.2,0.6	0.6841
0.9,0.9	0.6835
0.7,0.9	0.6874
0.5,0.9	**0.6918**

   The best MAP score is achieved for **k1 = 0.5**, **b= 0.9**.

2. Tweaking the parameters of the DFR Model

   Experimenting with various combinations of possible values for the parameters. Some of the combinations are as follows:

PARAMETERS	**MAP VALUE**
Be,B,H1	0.6939
G,B,H1	0.6942
P,B,H1	0.6908
D,B,H1	0.6949
I(n),B,H1	0.6928
I(ne),B,H1	0.6957
I(F),B,H1	**0.6964**

   . The best MAP is achieved for the following values of the parameters.
   Basic Model: **I(F)** (Inverse term frequency)
   After Effect: **B** (Ratio of two Bernoulli processes)
   Normalization[c]: **H1[7]** (Uniform distribution of term frequency with hyper-parameter that controls tf normalization with respect to document length)

3. Search the query in all the language fields.

   When we search documents for a given query, if we search in a particular language field, we may miss out on relevant documents of other languages. So we can use copyfield for this purpose. What we did was copy all the language fields into one separate field and queried the copied field.

4. Setting the slop values using the Dismax Query Parser (Not included in Schema)

   Analyzing the query results, it seems intuitive that the results having higher proximity of query terms are more relevant than the others.
   We can set this proximity by setting the ps (phrase slop) and qs (query slop) parameters using the Dismax Query Parser.

   This did not affect the overall map value of the system (20 rows). However, we kept it in the schema because the intuition behind using it seems legit and it may affect the map value for other queries.

5. Boosting some fields using the Dismax Query Parser.

   While querying something it is not necessary that all the fields being queried have equal importance. For example, if the query is in English it should be queried across the fields of all the languages but the results from the English field should rank higher than the others. This can be done by copying all fields into one field and querying all the fields but giving a boost to the individual language fields. The boost to the fields can be specified using the pf and qf fields in the Dismax Query Parser.

   Pf: text_en^2 text_ru^2 text_de^2

6. Query Expansion using Synonym Filter

   In this technique we used the SynonymFilterFactory to specify some of the general mapping based on the corpus given. This is done to increase the number of relevant results returned. These mappings have to be done in synonyms.txt. For example, (army, military) would map army to military and vice versa, so whenever a query is made results with both army and military thus returning more relevant results and thus increasing the map score.

   However, for the corpus provided, the use of this filter for query expansion only increased the map score for VSM andBM25 model but not for the DFR model. So we implemented this technique only for VSM and BM25 model.

   Code snippet:

   <filter class="solr.SynonymFilterFactory" expand="true" ignoreCase="true" synonyms="synonyms.txt"/>

7. Boosting important terms in a query (Not included in Schema)

   A query can be a single term or it can contain multiple terms. In the latter case each term of the query does not have the same importance. Thus the relevance of the documents returned should depend on the importance of the terms in the query. For example, in the first query 'Russia's intervention in Syria', the query term 'intervention' is more relevant than the other terms. So giving a boost to this term results in more relevant documents returned in the top documents and thus an increased map score. However, it should be kept in mind that some importance should also be given to the other terms in the query or the number of relevant documents will decrease, thus affecting the recall. So the resultant query looks something like 'Russia's^2 intervention^4 in Syria^2'.

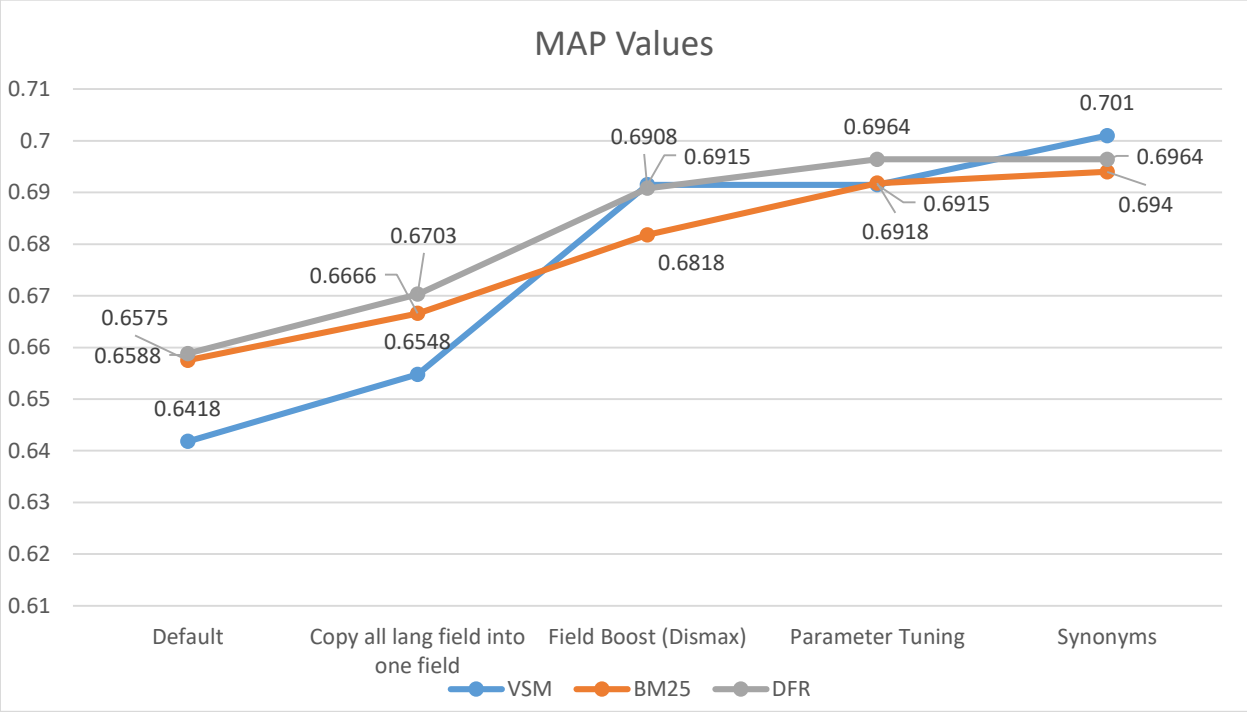
   However, this is based on query specific boosting and it cannot be used to improve the overall system and hence this was not included.

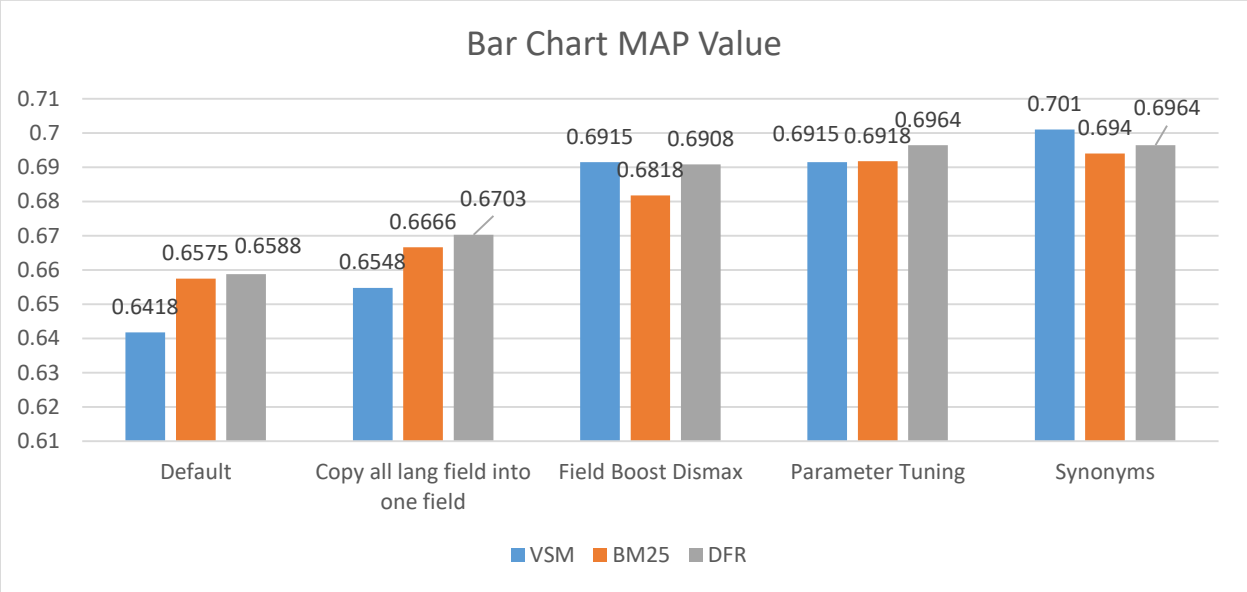
8. Query Elevation Component (Not included in Schema)

   The QueryElevationComponent enables you to configure the top results for a given query regardless of the normal lucene scoring. This is sometimes called "sponsored search", "editorial boosting" or "best bets".

   However, this is again based on query specific boosting and it cannot be used to improve the overall system and hence this was not included.

MAP Result After Each Step:





References:

- http://lucene.apache.org/solr/6_0_0/solr-core/org/apache/solr/search/similarities/ClassicSimilarityFactory.html
- http://lucene.apache.org/solr/6_0_0/solr-core/org/apache/solr/search/similarities/BM25SimilarityFactory.html
- http://lucene.apache.org/solr/6_0_0/solr-core/org/apache/solr/search/similarities/DFRSimilarityFactory.html
- https://wiki.apache.org/solr/SolrRelevancyFAQ
- http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-ranked-retrieval-results-1.html
- https://cwiki.apache.org/confluence/display/solr/The+DisMax+Query+Parser#TheDisMaxQueryParser
- https://cwiki.apache.org/confluence/display/solr/Filter+Descriptions
- Apache Solr Cookbook
- Solr 1.4 Enterprise Search Server
- http://journal.code4lib.org/articles/7787/comment-page-1
- https://cwiki.apache.org/confluence/display/solr/The+Extended+DisMax+Query+Parser
- http://nlp.stanford.edu/IR-book/html/htmledition/okapi-bm25-a-non-binary-model-1.html