PROJECT REPORT

starTREC

Ahut Gupta
Shaleen Mathur

Introduction

In this report, we evaluate 3 different IR models- Vector Space Model, Okapi BM 25 and Language Model. All the models were implemented in Apache Solr. We indexed all the models with the same twitter data, in three languages: English, German and Russian. A set of 14 queries was executed against each model and then their performances were measured and compared using TREC_EVAL, using five parameters: Total number of relevant documents retrieved, MAP, nDCG, BPREF and F0.5. Various performance tuning measures were also tried, to increase the TREC_EVAL scores, depending on each query, in each model.

Data Set

The data set consists of twitter data, collected in 3 different languages, English, German and Russian, on the topic of European Refugee Crisis. All the data was in json format with the following fields:

```
{
"lang": ,
"id": ,
"text_de": ,
"text_en": ,
"text_ru": ,
"created_at": ,
"tweet_urls": [ ],
"tweet_hashtags": []
}
```

Evaluation Measures used

1. **MAP(**Mean Average Precision):
   The MAP value for a test collection is the arithmetic mean of average precision values for individual information needs.

$$\mathrm{MAP}(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j}\mathrm{Precision}(R_{jk})$$

2. **nDCG**(normalized discounted cumulative gain):
   nDCG is measures of cumulative gain, designed for situations of non-binary notions of relevance. it is evaluated over some number **k** of top search results. For a set of queries **Q** , let **R(j, d)** be the relevance score assessors gave to document **d** for query **j**. Then,

$$\mathrm{NDCG}(Q,k) = \frac{1}{|Q|}\sum_{j=1}^{|Q|} Z_{kj} \sum_{m=1}^{k} \frac{2^{R(j,m)}-1}{\log_2(1+m)},$$

   where Z_{kj} is a normalization factor.

3. **BPREF**:
   The bpref measure is designed for situations where relevance judgments are known to be far from complete. It was introduced in the TREC 2005 terabyte track. BPREF computes a preference relation of whether judged relevant documents are retrieved ahead of judged irrelevant documents.

$$bpref = \frac{1}{R}\sum_{r}\left(1 - \frac{|n \text{ ranked higher than } r|}{min(R,N)}\right)$$

4. **F0.5:**
   This is a measure that combines precision and recall, and is the harmonic mean of both.

$$F = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

Models Implemented

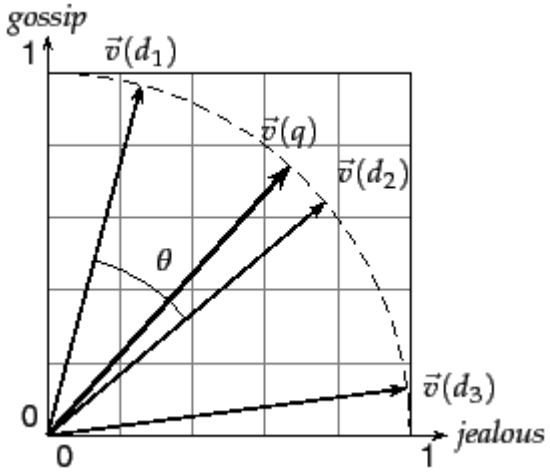
I. Vector Space Model

Introduction

In this model, we treat queries as vectors and all the documents and queries are represented in this common vector space. The similarity between any two vectors in the vector space can be determined from their Cosine similarity score. Which is defined as:

$$\text{cosine-similarity}(q,d) = \frac{V(q) \cdot V(d)}{|V(q)|\,|V(d)|}$$

Here, the numerator represents the dot product of the query and document vectors and, while the denominator is the product of their Euclidean lengths.



The practical score computed by Solr, is based on **tf** (term frequency) and **idf** (inverse document frequency). Where,

$$tf(t \text{ in } d) = \text{frequency}^{½}$$

$$idf(t) = 1 + \log\left(\frac{\text{numDocs}}{\text{docFreq}+1}\right)$$

Implementation

We implemented VSM with the default similarity class in lucene.

```
<similarity class="org.apache.lucene.search.similarities.DefaultSimilarity"/>
```

This class was part of the schema.xml file and implements the default similarity, which extends TFIDF similarity, which implements the Vector Space Model.

Tuning

- ## Multi-Lingual search

  Since we have data in 3 languages, it makes sense to search any query against all the language text fields. We performed tests on Query # 5 in the training data set for this analysis.

  For the base case, we searched a query only against the text field of the language of the query. In this case- Russian.

  q= text_ru:(РФ в Сирии вынудили 250 тунисских боевиков бежать)

  Scores:

map	0.5000
ndcg	0.7244
F.0.5	0.0088

  Case:1- Search the same text(original language) in all the text fields.
  q= text_de:(РФ в Сирии вынудили 250 тунисских боевиков бежать) OR text_en:(РФ в Сирии вынудили 250 тунисских боевиков тунисских боевиков бежать) OR text_ru:(РФ в Сирии вынудили 250 тунисских боевиков бежать)
  Scores:

map	0.5075
ndcg	0.7689
F.0.5	0.0131

  Case:2- Multilingual search. Search translated texts in all the fields.
  q= text_ru:(РФ в Сирии вынудили 250 тунисских боевиков бежать) OR text_en:(Russian Federation in Syria forced the militants to flee Tunisia 250) OR text_de:(Russland in Syrien gezwungen, die Militanten nach Tunesien 250 fliehen)

  Scores:

map	0.6886
ndcg	0.8820
F.0.5	0.0064

  It is clear from the above scores that we obtain a definite increase in performance when we did a multilingual search. Therefore, we follow the same pattern for all the subsequent analyses.

- **Query Boosting**

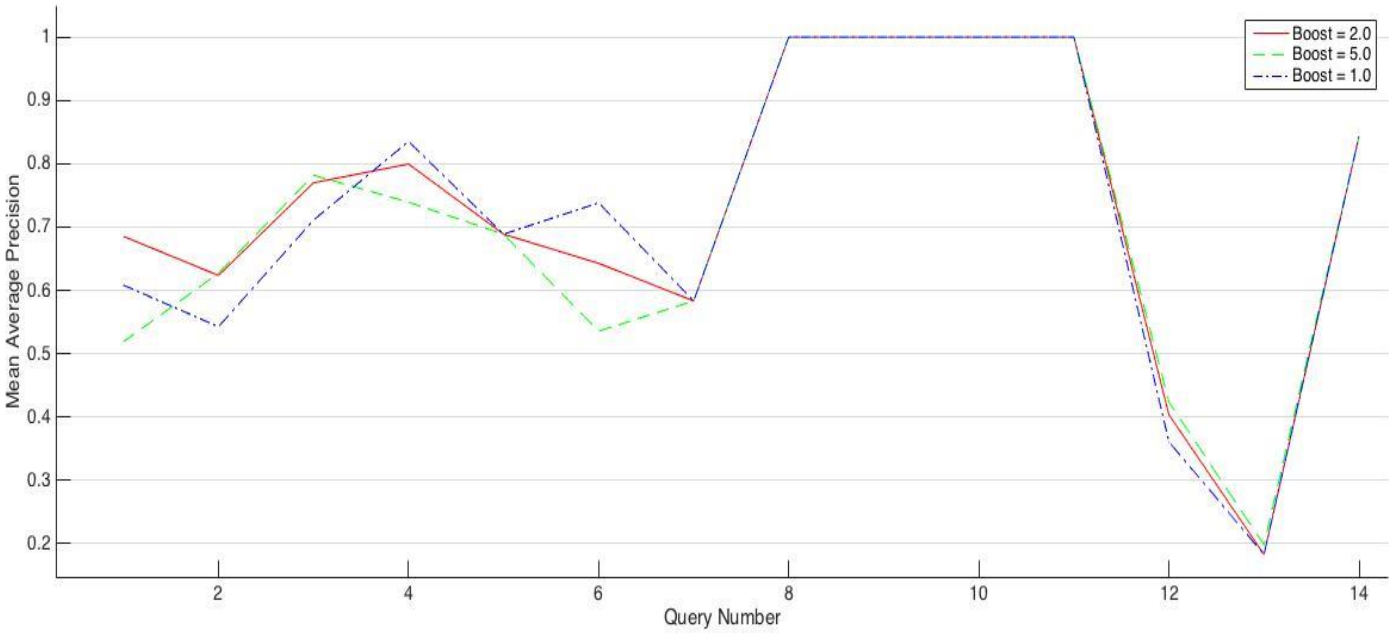
  We considered the MAP parameter for query boosting. Queries were translated and searched across all the languages. And we boosted the text field for the language of the query.

  Eg- For Query #2 in the training queries, the query is: “US air dropped 50 tons of Ammo on Syria”. The original language of the query is English. We translated the query text into German and Russian, and searched the fields as:

  text_en:(US air dropped 50 tons of Ammo on Syria) OR text_de:(US Luft abgeworfen 50 Tonnen Munition zu Syrien) OR text_ru:(США воздуха упала 50 тонн боеприпасов на Сирию)

  Since the original query was in English, we boosted the field: “text_en” with boost values of 1, 2 and 5.

  We compiled a list of scores for all the queries and plotted them against the various boost values. The following graph was obtained:



Boost scores for all queries.

Boost	MAP
1	0.721
2	0.730
5	0.710

As we can see, the highest scores were obtained for **boost value of 2.**

Therefore, we use a boost of 2 for all the subsequent analyses.

- **Term Weights**

  In a query, some terms are more important than the others. Which means if we can increase the score of documents where those terms match, we get an overall increase in the performance.
  Since the documents in our collection have a maximum length of 140 characters, any **Proper Nouns** in the query which match a document will be the subject of that document and hence make there is a high chance that the corresponding document is relevant to the query.
  We experimented with the standard query parser, by adding term weights for proper nouns in the query.

  Eg- For query #4 in the training queries, we followed the multilingual search as described above, with term weight of 10 for the various nouns.

  Base Case: No term weights.

  q= text_de:(Wegen Flüchtlingskrise Angela Merkel stürzt in Umfragen) OR text_en:(Because refugee crisis Angela Merkel crashes in Polls) OR text_ru:(Потому что кризис беженца Ангела Меркель падает в опросах)

  Scores:

map	0.7613
bpref	0.8357
ndcg	0.9659
F.0.5	0.0391

  With weights

  text_de:(Wegen Flüchtlingskrise^10 Angela^10 Merkel^10 stürzt in Umfragen) OR text_en:(Because refugee^10 crisis Angela^10 Merkel^10 crashes in Polls) OR text_ru:(Потому что кризис беженца^10 Ангела^10 Меркель^10 падает в опросах)

  Scores:

map	0.8110
bpref	0.8357
ndcg	0.9763
F.0.5	0.0391

  It is clear that using term weights for nouns in the query, increases the scores. Hence we use similar term weights for all the subsequent analyses.

II. BM-25 Model

Introduction

BM 25 is a probabilistic information retrieval model. A relevance score is calculated, based on probabilistic concepts, reflect the probability a user will consider the result relevant. This score is based on term frequency, document frequency, document lengths and 2 tuning parameters: k1 and b.

$$RSV_d = \sum_{t \in q} \log \left[\frac{N}{\mathrm{df}_t} \right] \cdot \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1-b) + b \times (L_d / L_{ave})) + \mathrm{tf}_{td}}$$

The variable k1 is a positive tuning parameter that calibrates the document term frequency scaling. A k1 value of 0 corresponds to a binary model (no term frequency), and a large value corresponds to using raw term frequency. b is another tuning parameter ($0 < b < 1$) which determines the scaling by document length: b = 1 corresponds to fully scaling the term weight by the document length, while b = 0 corresponds to no length normalization.

Implementation

We implemented BM25 model with the BM25 similarity factory in solr.

<similarity class="org.apache.solr.search.similarities.BM25SimilarityFactory">

This class was part of the schema.xml file and implements the BM25 similarity factory, which extends the Similarity Factory class in solr.

Tuning

- **K1 and b parameters in the model**
  BM25 similarity factory in lucene provides 2 parameters: k1 and b for the IR model. We tweaked the values of these parameters, based on query #6.

  Base Case(Default values):
  K1= 1.2 and b= 0.75 (default values in the similarity factory)

  Scores:

map	0.7371
ndcg	0.9068
F.0.5	0.0351

Case:1- Modifying b.
K1= 1.2 and b= 0.1
Scores:

map	0.7057
ndcg	0.9018
F.0.5	0.0351

Case:2- Modifying b.
K1= 1.2 and b=1
Scores:

map	0.8164
ndcg	0.9385
F.0.5	0.0351

Based on the scores of these 2 cases, and the fact that the (0<b<1),
We can conclude that the **optimum value for b is 1.**

Case:3- Modifying k1.
K1= 2.3 and b=1
Scores:

map	0.8201
ndcg	0.9395
F.0.5	0.0351

Case:4- Modifying k1.
K1= 5 and b=1
Scores:

map	0.8131
ndcg	0.9377
F.0.5	0.0351

Based on cases 2,3 and 4, we see that we achieve the highest scores for k1=2.3.
Therefore, we take the **optimum value of k1 as 2.3.**

- **Multi-lingual Search**

  As described in the above model (VSM), we implemented multilingual search on this model as well.

  Eg-1: Query #1.
  Base Case:

  q=text_ru:(РФ в Сирии вынудили 250 тунисских боевиков бежать)

  Scores:

map	0.500
ndcg	0.8357
F.0.5	0.0088

  With Multilingual search:

  q= text_ru:(РФ в Сирии вынудили 250 тунисских боевиков бежать) OR text_en:(Russian Federation in Syria forced the militants to flee Tunisia 250) OR text_de:(Russland in Syrien gezwungen, die Militanten nach Tunesien 250 fliehen)

  Scores:

map	0.500
bpref	0.8357
F.0.5	0.0088

  We observe no difference in the scores for both the cases. Therefore we can conclude that doing a multilingual search in BM25 does not affect the performance.

- **Phrase Matching**

  We implemented phrase search in the standard query parser.
  We pick **query #1** for this analysis, as it is a short query and phrase matching has a higher chance of giving better results, if we parse the short queries as a single phrase.

  Base Case:
  Q=text_en: (Russia's intervention in Syria)

Score:

map	0.500
bpref	0.8357
F.0.5	0.0088

Case:1- Phrase matching

Q= text_en:(Russia intervention Syria)^2 || text_en:"Russian intervention in Syria"

Score:

map	0.5267
bpref	0.8875
F.0.5	0.0980

It is clear from the above scores that phrase matching increases the performance of the IR model. Therefore we can use phrase matching for all the short queries.

- **Query Expansion using Synonyms**

We tried query expansion for query #5, by adding words in the synonyms list.
Base case: No Synonyms
Multilingual search.
Score:

map	0.5000
bpref	0.7244
F.0.5	0.0088

Case:1- Following entries in the synonyms.txt file:
ISIS, ISIL
militants, fighters, mercenaries
slaughter, forced

Multilingual search.
Score:

map	0.6933
bpref	0.8917
F.0.5	0.0064

We see that the scores improved after adding synonyms. Therefore we will add use this synonyms file for all the future analyses.

III. Language Model

Introduction

Language model is a probability distribution over sequences of words. Given such a sequence, say of length m, it assigns a probability P(w_1,……,w_m) to the whole sequence. It follows the approach that: a document is a good match to a query if the document model is likely to generate the query, which will in turn happen if the document contains the query words often. Documents are ranked based on the probability of the query *Q* in the document's language model . $P(Q \mid M_d)$

$$p(w \mid d) = \begin{cases} p_s(w \mid d) & \text{if word } w \text{ is seen} \\ \alpha_d p(w \mid C) & \text{otherwise} \end{cases}$$

The Dirichlet method. This method involves a linear interpolation of the maximum likelihood model with the collection model, using a coefficient λ to control the influence of each:

$$p_\mu(w \mid d) = \frac{c(w;d) + \mu\, p(w \mid C)}{\sum_{w' \in V} c(w';d) + \mu}$$

Implementation

We implemented Language model with 2 different similarity classes in lucene.

1. LMDirichletSimilarity
2. LMDirichletSimilarityFactory

Based on all the above optimizations, we executed all the queries for both the above implementation. The key techniques we used were:

a. Multilingual search
b. Query boosting in the corresponding language field.
c. Query expansion using synonyms.
d. Phrase matching for small queries.

We obtained the following scores:

1. LMDirichletSimilarity

   map all 0.7365

   ndcg all 0.9155

   set_F_0.5 all 0.2540

2. LMDirichletSimilarityFactory

   map all 0.7336

   ndcg all 0.9151

   set_F_0.5 all 0.2532

Tuning

Solr provides a parameter (mu) for the Language IR model.
We experimented with various values of mu to obtain the best result.

1. Mu = 1000

map all 0.7336

ndcg all 0.9151

F_0.5 all 0.2532

2. Mu = 0

map 002 0.1279
ndcg 002 0.4199
set_F_0.5 002 0.0818

3. Mu = 1

map 002 0.6420

ndcg 002 0.8433

set_F_0.5 002 0.0818

4. **Mu = 100**

   map 002 0.3902

   ndcg 002 0.7326

   set_F_0.5 002 0.0818

5. **Mu = 10000**

   map 002 0.3225

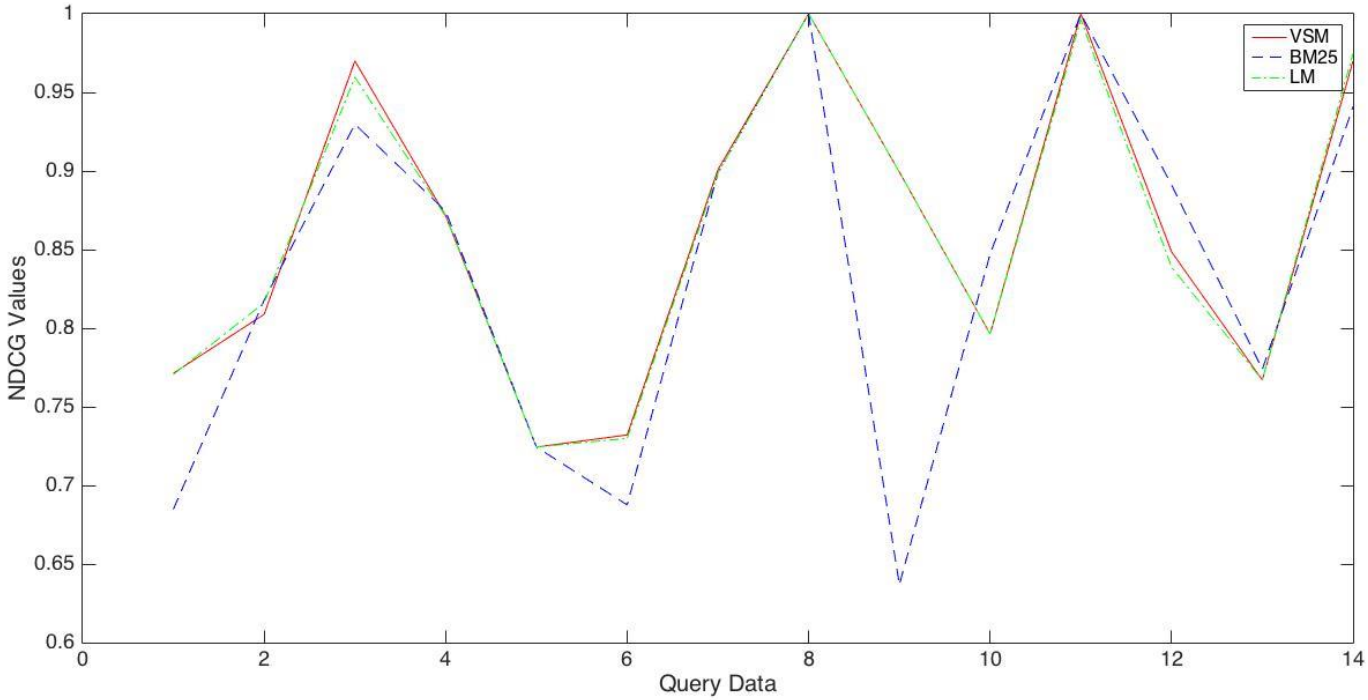
   ndcg 002 0.6923

   set_F_0.5 002 0.0818

Conclusion

Which Model performs better

We plot a graph of all the ndcg scores of all the queries for all the three models.



From the above graph we observe that we get higher nDCG values across most of the queries for the Vector Space Model.

Hence, we can conclude that **Vector Space Model is the best among the three.**

Which measure gives a better evaluation

Mean Average Precision (MAP), which provides a single-figure measure of quality across recall levels. Among evaluation measures, MAP provides good discrimination and stability.

nDCG is a normalized measure. Since, for this data set, all the documents are very short, i.e. max length of 140 characters in tweets, Normalization does not affect the performance much.

Therefore we can say that **MAP is a better measure of performance for the 3 models we have implemented.**

References:

- http://nlp.stanford.edu/
- http://trec.nist.gov/
- https://en.wikipedia.org/wiki/
- http://www.stat.uchicago.edu/~lafferty/pdf/smooth-tois.pdf