Evaluation of IR Models

By

Hrishikesh Saraf (hsaraf)

Nikita Konda (nkonda)

The goal of this project is to implement various models, evaluate their performance and improve our IR system. The models that we implemented under this project are Vector Space Model, BM25 Model and Divergence from Randomness Model.

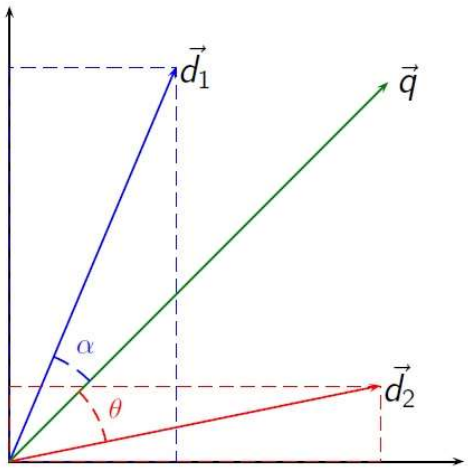
Vector Space Model

Vector space model or term vector model is an algebraic model for representing text documents as vectors of identifiers, such as, for example, index terms. [1]

The documents and queries are represented in the form of vectors:

$d_j = (w_{1,j}, w_{2,j}, \ldots w_{t,j})$

$q = (w_{1,q}, w_{2,q}, \ldots w_{t,q})$



Relevance rankings of documents is calculated, by comparing the deviation of angles between each document vector and the original query vector where the query is represented as the same kind of vector as the documents.

$$\cos\theta = \frac{\mathbf{d_2} \cdot \mathbf{q}}{\|\mathbf{d_2}\| \, \|\mathbf{q}\|}$$

Where $d_2.q$ is the intersection of the documents is the intersection (the dot product) of the document and the query vectors, $\|d_2\|$ is the norm of vector d2, and $\|q\|$ is the norm of vector q. The norm of a vector is calculated as such:

$$\|\mathbf{q}\| = \sqrt{\sum_{i=1}^{n} q_i^2}$$

We used the ClassicSimilarity class of Solr which extends TFIDF similarity that implements similarity with the Vector Space Model.

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

The value of MAP (mean-average-precision) that was evaluated without any changes to the schema (default value) was **0.6347**.

Improvements:

Usage of Synonym Filter:

To make sure that the query that we use fetches documents from all three languages (German, English and Russian) we used the synonym filter with synonyms from all languages for every significant keyword. This filter does synonym mapping. Each token is looked up in the list of synonyms and if a match is found, the synonym is emitted in place of the token. The position value of the new tokens are set such that they all occur at the same position as the original token [2].

To optimize the results of synonym filter for our IR system we used our own customized "synonyms.txt" with keywords and synonyms relevant to the given topic.

New MAP value: **0.6531**

Query Parsing:

We tried Standard, DisMax and eDisMax query parsers. But out of them, the best results were achieved using DisMax query parser.

The DisMax query parser is designed to process simple phrases (without complex syntax) entered by users and to search for individual terms across several fields using different weighting (boosts) based on the significance of each field [3]. Under DisMax query parser we have different parameters that we can specify such as qf.

qf: Query Fields: specifies the fields in the index on which to perform the query [3].

We used the DisMax query parser to boost the fields and after several trials we added weights as following:

text_en^15 + text_ru^8 + text_de^15 + tweet_hashtags^5 + tweet_urls^5

MAP value after query parsing: **0.7028**

VSM Screenshot:

```
runid                   all     VSM
num_q                   all     20
num_ret                 all     380
num_rel                 all     305
num_rel_ret             all     174
map                     all     0.7028
gm_map                  all     0.6372
Rprec                   all     0.6940
bpref                   all     0.7061
```

BM25 Model

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). [1]

The default similarity class in solr-6.2.0 is BM25. If no similarity class is added to the schema, the similarity class that is used by default is BM25SimilarityFactory. However, we can add BM25SimilarityFactory class to the schema to tweak the values of k1 and b.

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

The default MAP score with BM25 is **0.6575.**

Improvements:

Tweaking Parameters of BM25 Model:

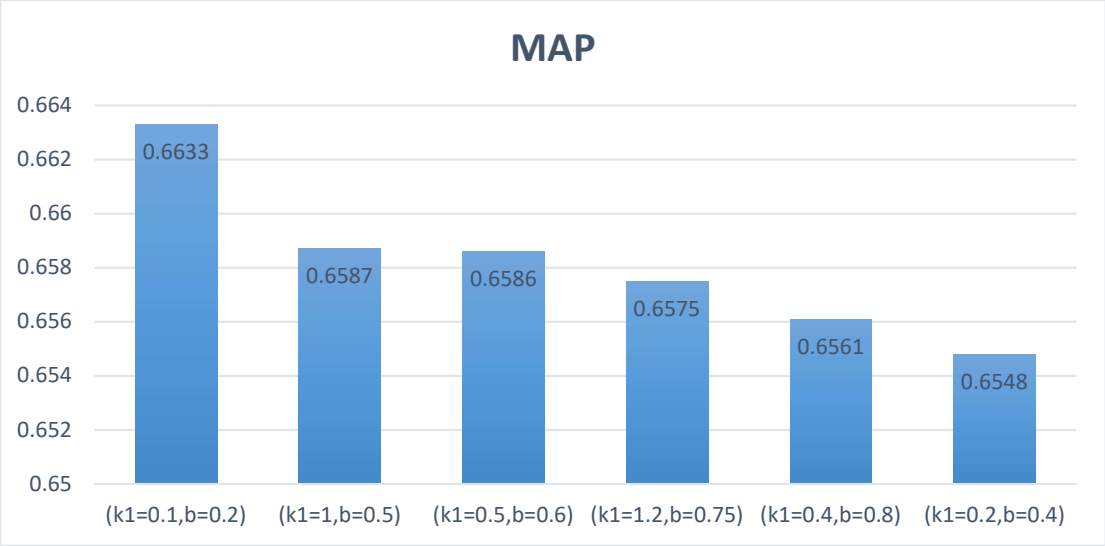
Given a query Q, containing keywords q_1, q_2…..q_n the BM25 score of a document D is[1]:

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

where $f(q_i, D)$ is q_i's term frequency in the document D, |D| is the length of the document D in words, and avgdl is the average document length in the text collection from which documents are drawn.k_1 and b are free parameters, usually chosen, in absence of an advanced optimization, as $k_1 \in [1.2, 2.0]$ and b = 0.75. $\text{IDF}(q_i)$ is the IDF (inverse document frequency) weight of the query term q_i. It is usually computed as:

$$\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

Following is a graph that represents MAP values for different values for parameters k_1 and b.



Finally, we select $k_1 = 0.1$ and $b = 0.2$ as the ideal set of parameters.

Query Parsing:

We tried Standard, DisMax and eDisMax query parsers. But out of of three, the best results were achieved using DisMax query parser. We used the DisMax query parser to boost the fields and after several trials we added weights as following:

text_en^15 + text_ru^8 + text_de^15 + tweet_hashtags^5 + tweet_urls^5

MAP score after field boosting: **0.6918**

Usage of UAX29URLEmailTokenizerFactory:

We tried default, Classic, Keyword, Letter, Lower Case and UAX29URLEmailTokenizerFactory and found that UAX29URLEmailTokenizerFactory gives the best result.

MAP value after usage of UAX29URLEmailTokenizerFactory: **0.6985**

BM25 Screenshot:

```
runid                   all     BM25
num_q                   all     20
num_ret                 all     380
num_rel                 all     305
num_rel_ret             all     175
map                     all     0.6985
gm_map                  all     0.6305
Rprec                   all     0.6933
bpref                   all     0.7034
```

Divergence from Randomness Model:

In Divergence from Randomness Model (DFR) the term weights are computed by measuring the divergence between a term distribution produced by a random process and the actual term distribution [1].

For the implementation of DFR we used the DFRSimilarityFactory class in Solr. The DFRSimilarityFactory has three parameters components: basicModel, afterEffect, normalization. There are several implementations of the three components as follows[4]:

1. basicModel: Basic model of information content:
    - Be: Limiting form of Bose-Einstein
    - G: Geometric approximation of Bose-Einstein
    - P: Poisson approximation of the Binomial
    - D: Divergence approximation of the Binomial
    - I(n): Inverse document frequency
    - I(ne): Inverse expected document frequency [mixture of Poisson and IDF]
    - I(F): Inverse term frequency [approximation of I(ne)]
2. afterEffect: First normalization of information gain:
    - L: Laplace's law of succession
    - B: Ratio of two Bernoulli processes
    - none: no first normalization
3. normalization: Second (length) normalization:
    - H1: Uniform distribution of term frequency
        - parameter c (float): hyper-parameter that controls the term frequency normalization with respect to the document length. The default is 1
    - H2: term frequency density inversely related to length
        - parameter c (float): hyper-parameter that controls the term frequency normalization with respect to the document length. The default is 1
    - H3: term frequency normalization provided by Dirichlet prior
        - parameter mu (float): smoothing parameter μ. The default is 800
    - Z: term frequency normalization provided by a Zipfian relation
        - parameter z (float): represents A/(A+1) where A measures the specificity of the language. The default is 0.3
    - none: no second normalization

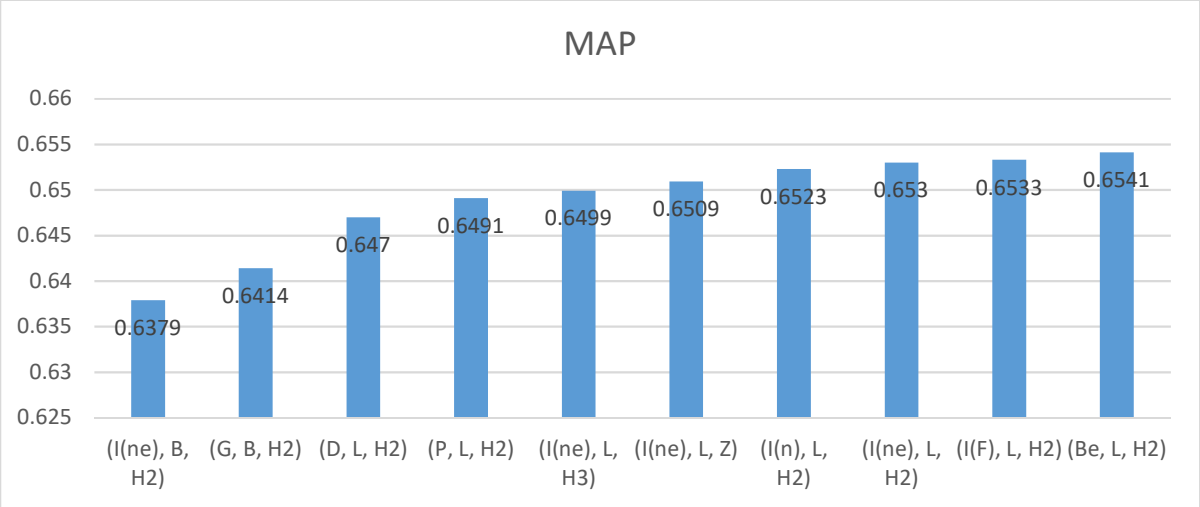
First we implemented the DFR model on given implementations of basicModel: G, afterEffect: B and normalization: H2 and we found the MAP value to be **0.6554**.

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

Improvements:

Tweaking parameters of the model:

We tweaked the different parameters of the BM25 model. Following is a Parameters vs MAP graph for all the trials performed.



After several trials, the implementations that we found optimum for our IR system were basicModel: Be , afterEffect: L and normalization: H2 with MAP value **0.6541**.

Query Parsing:

We tried Standard, DisMax and eDisMax query parsers. But out of the three, the best results were achieved using DisMax query parser. We used the DisMax query parser to boost the fields and after several trials we added weights as following:

text_en^15 + text_ru^8 + text_de^15 + tweet_hashtags^5 + tweet_urls^5

MAP score after field boosting: **0.7005**

DFR Screenshot:

```
runid                   all     DFR
num_q                   all     20
num_ret                 all     380
num_rel                 all     305
num_rel_ret             all     177
map                     all     0.7005
gm_map                  all     0.6402
Rprec                   all     0.6912
bpref                   all     0.7081
```

Conclusions:

Based on all the above improvements and changes made to the IR models, we concluded that the Vector Space Model (MAP: 0.7028) works best for the given training set.

References:

[1]: Wikipedia

[2]: https://cwiki.apache.org/confluence/display/solr/Filter+Descriptions#FilterDescriptions-SynonymFilter

[3]: https://cwiki.apache.org/confluence/display/solr/The+DisMax+Query+Parser

[4]: http://lucene.apache.org/solr/5_5_0/solr-core/org/apache/solr/search/similarities/DFRSimilarityFactory.html