**IR PROJECT3 REPORT**

**Team Members:**

Tariq Siddiqui

Kaushik Ramasubramanian

**Below are the default settings, on top of which we start our Implementation:**



**Result Summary:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Default Setting** | **After Language Translation** | **Query Weight Boosting** | **After**  **Synonyms** | **URL Removal** | **Model Specific Tuning** |
| VSM | 0.6469 | 0.6520 | 0.7030 | 0.7079 | 0.7271 | 0.7271 |
| DFR | 0.6468 | 0.6887 | 0.7082 | 0.7078 | 0.7132 | .7200(G, L and H1) |
| BM25 | 0.6554 | 0.6669 | 0.6873 | 0.7068 | 0.7174 | .7200(K1=1,b=.55) |

**1. Various Model Implementation**

Different IR Similarity models are implemented on different cores in solr for each model.

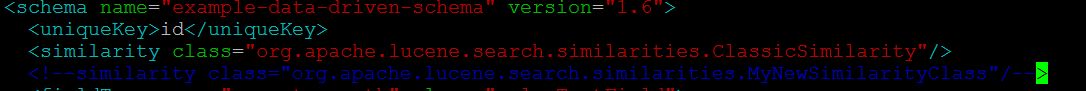
To run in Schema Mode, we have used schema.xml (instead of Managed-schema which is used in schema less mode) to define various filters, fields, field types and the similarity implementation for each model.

* 1. **Vector Space Model (ClassicSimilarity Model)**

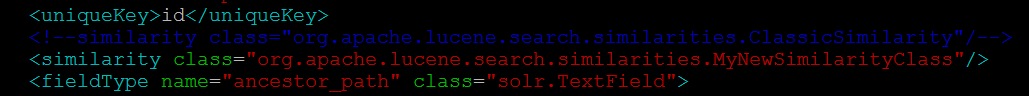
It is a type of IR model used to represent the text documents as vectors of identifiers. In this documents and queries are both vectors.

This model uses the term frequency (tf) and inverse document (idf) to calculate the weight vector of the document.

**Wt,d = tf \* log(n/df)**



*Figure 1 : schema.xml for VSM* ***Default*** *Similarity implementation*

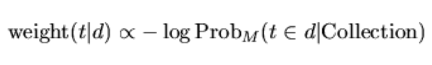


*Figure 2: schema.xml for VSM* ***Custom*** *Similarity class implementation*

* 1. **DFR MODEL**

The more the divergence of the within-document term-frequency from its frequency within the collection, the more the information carried by the word t in the document ‘d’.

The term-weight is inversely related to the probability of term-frequency within the document d obtained by a model M of randomness:



where the subscript M stands for the type of model of randomness employed to compute the probability.

In order to choose the appropriate model M of randomness, we can use different urn models. IR is thus seen as a probabilistic process, which uses random drawings from urn models. There are many ways to choose M, each of these provides a basic DFR model. These basic models are:

D Divergence approximation of the binomial

P Approximation of the binomial

BE Bose-Einstein distribution

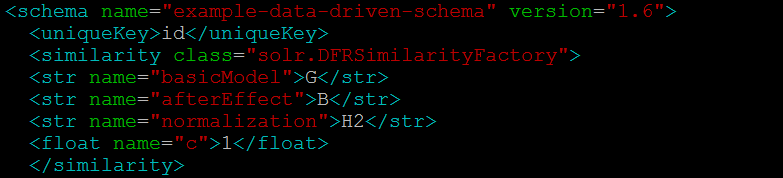
G Geometric approximation of the Bose-Einstein

I(n) Inverse Document Frequency model

I(F) Inverse Term Frequency model

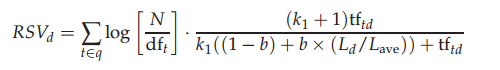
I(ne) Inverse Expected Document Frequency model

As mentioned in Section5 of Project Guide, For the DFR model, we have chosen ‘BasicModelG’as the basic Model, ‘Bernoulli First Normalization’ as after Effect parameter and ‘H2’ second normalization for normalization parameter.

 *Figure 3: Schema.xml for DFR* ***Default*** *Similarity implementation*

* 1. **BM25 Model**

The BM25 weighting scheme was developed as a way of building a probabilistic model sensitive to these quantities while not introducing too many additional parameters into the model.



In the above equation,

tftd is the frequency of term t in document d

Ld and Lave are the length of document d and the average document length for the whole collection

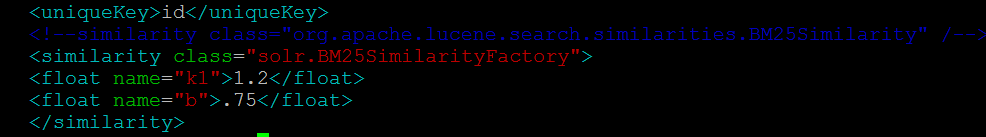
k1 is a positive tuning parameter that calibrates the document term frequency scaling

b is another tuning parameter

In order to get most relevant documents for a particular query**, it is required to have a greater value of ‘k1’ as it indicates raw term frequency**. **The value of ‘b’ ranges between 0 <= b <= 1. However, in order to get a relevant document, the value of ‘b’ has to be as small as possible.**

For BM25 model, we below show default BM25 settings which are K1= 1.2 and b =0.75

NOTE: Later we have played with k1 and b values. Below K1 and b values are default values we show for 1st 5 points.



*Figure 3: Schema.xml for BM25* ***Default*** *Similarity implementation*

|  |  |
| --- | --- |
| **Model** | **MAP Values for Default Setting** |
| VSM | 0.6469 |
| DFR | 0.6468 |
| BM25 | 0.6554 |

**2. Techniques used to improve Performance on Top of above default Values**

1. Query language translation
2. Field Specific Querying using above Translation

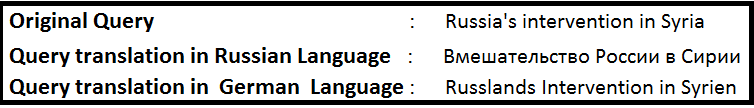
3(a) Boosting various fields using different weight

3(b) Boosting Text Field using Language of Query

1. Synonym Factory Implementation
2. Garbage Information(URL) Removal from Text Field
3. Exact Case matching (Using Copy Field)
4. Exact “Term” matching (Using synonym Factory and Copy Field)
5. Script Automation for BM25 and DFR Models for Model Specific optimization
6. Dismax for Phrase query
7. Customizing default Similarity Classes

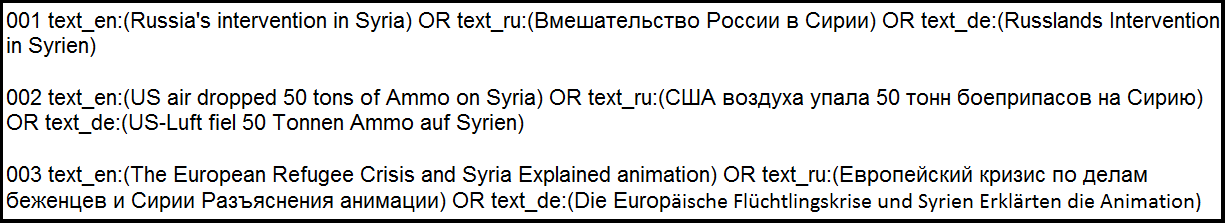
**2.1 Query language translation**

**Motivation:** We found that SOLR was not returning any Non-English documents for English Queries and Similar Observation was seen for other languages as well. Logically understanding this ambiguity, a Russian Twitter user is expected to write same content/Information in Russian language and hence, our implementation should search for same Information in Russian language as well.



**Implementation Impact:** Additionally, we realized it’s not Just sufficient to search blindly with all above translated queries but institutively it makes sense to search Russian query in Russian field, English in English and German and German. Therefore, we combine this information in Implementation 2 (explained below) with previous information and combined calculate map for Implementation 1 and 2 as below.

**2.2 Field Specific Querying using above Translation**



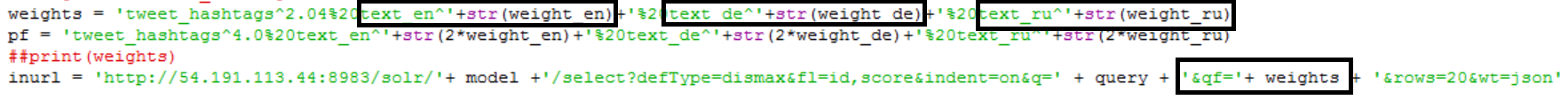
From the technique used in above section 2.1 we found that it is not only enough to translate the query language but it is also required to search the query in the same language in which it is required. Therefore, we have included an OR operation for all same query in all the three languages.

**Motivation:** To return the relevant documents for the query in the same language as that of the query.

|  |  |  |
| --- | --- | --- |
| **Model** | **Default Setting** | **After Language Translation** |
| VSM | 0.6469 | 0.6520 |
| DFR | 0.6468 | 0.6887 |
| BM25 | 0.6554 | 0.6669 |

**Table 1: Improvement is MAP values after Query Translation**

**2.3(a) Boosting various fields using different weight**



**Motivation:** All fields in a document don’t represent Information of Same priority. Example: Author, Head, Body, abstract both contain information with different relevance.

**Implementation:** Twitter data implicitly is categorized into components of varying information. Twitter mentions and twitter URLs represent very precise information. We exploit this logic.

Below we assigns weights to different fields such as ‘tweet\_hashtags’, ‘mentions’, ’tweet\_urls’ etc. with varying weights, that is assigning more weight to highly relevant fields and Low weight to less relevant fields .

More, we realized it’s not just important to Set weight but also weight depending on Language in which query was executed.

Edismax query parser is used to search across multiple fields with variable weightage. Dismax query parser allows to search across fields and add weights to each field.

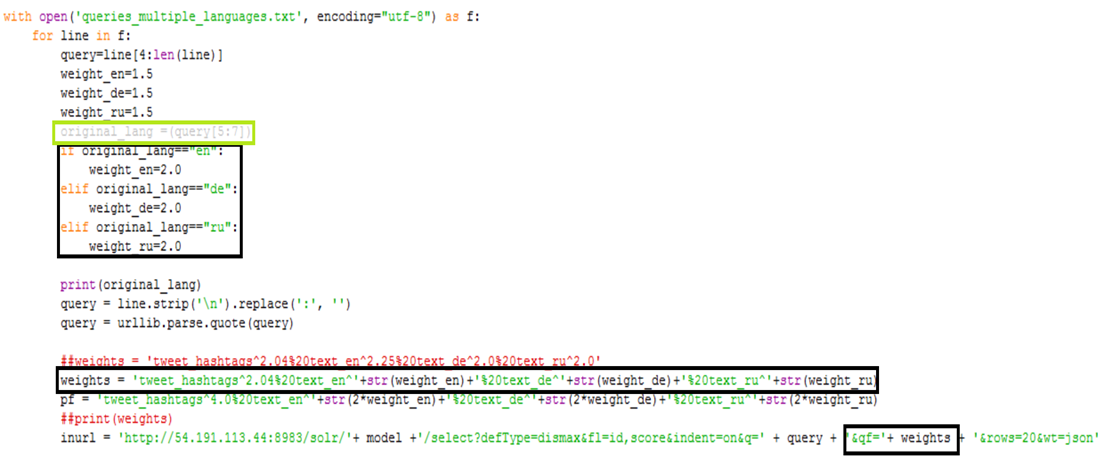
|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Default Setting** | **After Language Translation** | **Query Weight Boosting** |
| VSM | 0.6469 | 0.6520 | 0.7030 |
| DFR | 0.6468 | 0.6887 | 0.7082 |
| BM25 | 0.6554 | 0.6669 | 0.6873 |

**Table 2: Improvement in MAP values after weight boosting**

**2.3(b) Boosting Text Field using Language of Query**

**Motivation:** A query searched in particular language is more likely to be relevant to documents in that language. Hence, its becomes imperative to set weight of test field mode if Query is in language sa as that text field

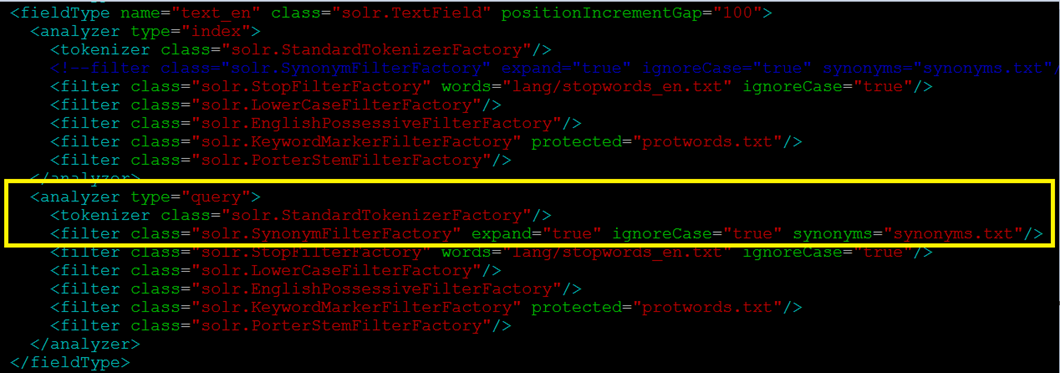
In this technique, as shown below we extract the language and then we set weight of text field with that language higher than other text fields.

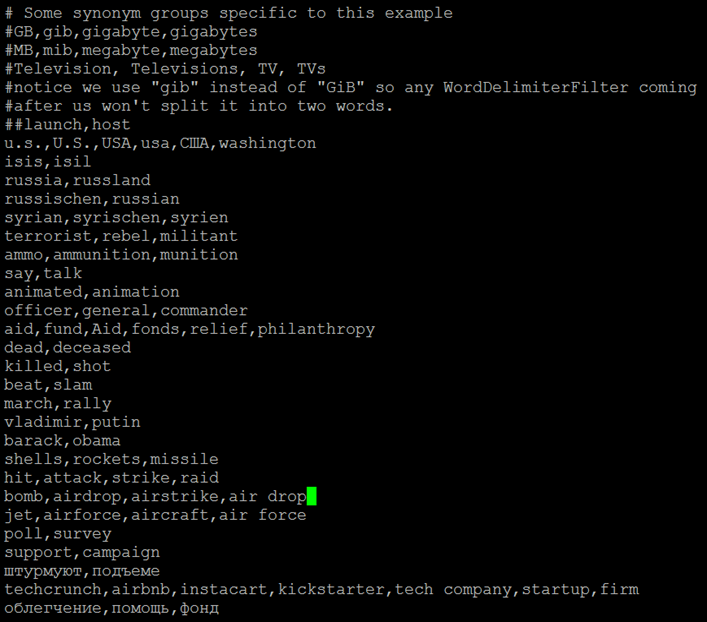


**2.4 Synonym Factory Implementation**

**Motivation:** Here,we increase recall through use of synonym factory which obviously match similar meaning words.

**Implementation changes**: Synonym factory implementation **is only applied to Query Field as below**. Applying same while indexing didn’t result is any significant gain.





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Default Setting** | **After Language Translation** | **Query Weight Boosting** | **After**  **Synonyms** |
| VSM | 0.6469 | 0.6520 | 0.7030 | 0.7079 |
| DFR | 0.6468 | 0.6887 | 0.7082 | 0.7078 |
| BM25 | 0.6554 | 0.6669 | 0.6873 | 0.7068 |

**Table 3: Improvement in MAP values after applying synonym factory**

**2.5 Garbage Information (example: URL) Removal from Text Field**

**Motivation:** Garbage content in the tweet reduces the precision.

**Implementation:** In this Implementation, garbage fields such as URL is removed from the test field.

After this implementation, we observe more precision search is done only on relevant data.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Default Setting** | **After Language Translation** | **Query Weight Boosting** | **After**  **Synonyms** | **URL Removal** |
| VSM | 0.6469 | 0.6520 | 0.7030 | 0.7079 | 0.7271 |
| DFR | 0.6468 | 0.6887 | 0.7082 | 0.7078 | 0.7132 |
| BM25 | 0.6554 | 0.6669 | 0.6873 | 0.7068 | 0.7174 |

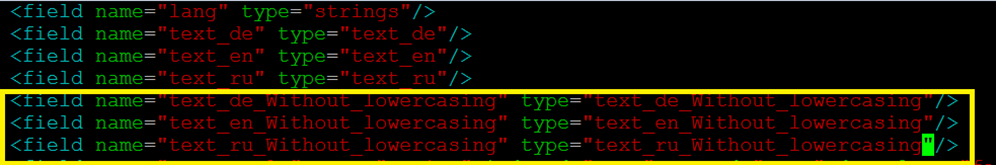
**Table 4: Improvement in MAP values after URL removal**

**2.6 Exact case matching (Using copy field)**

**Motivation:** More weight should be given to document which matches query along with Case as compared to document which matches query term but not Case.

As below, copy field of original text field is given more weight as compared to original field while performing query search operation.

We tried to implement this technique but couldn’t see any changes in MAP as the number of tweets in database is very less.





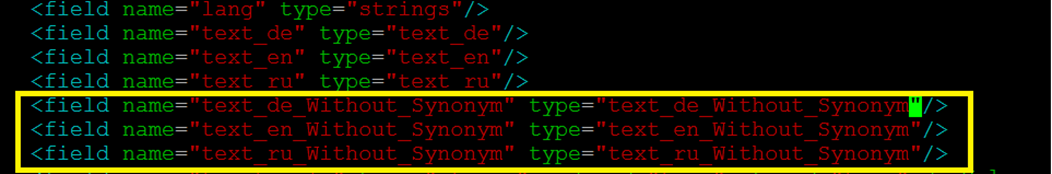


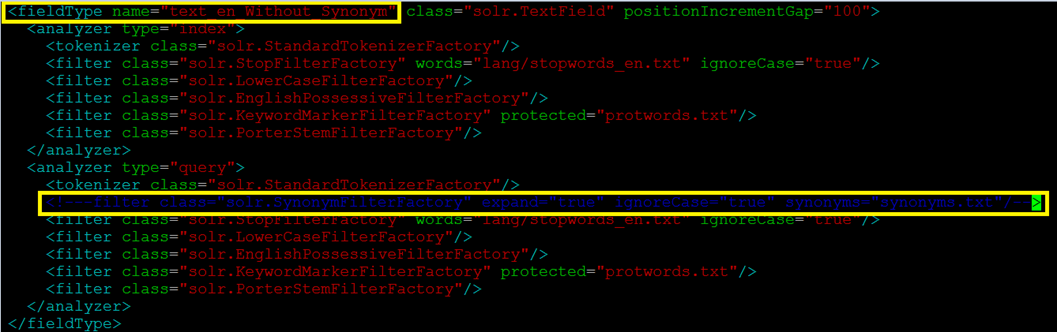
**2.7 Exact “Term” matching (Using synonym Factory and Copy Field)**

**Motivation:** More weight should be given to document which matches Exact query “Term” as compared to document which matches Synonym of query term.

To implement above, we make a copy field of the original text field, but disable synonym factory in this field and assign it a higher weight.

We tried to implement this but couldn’t see any changes in the MAP.





**2.8 Script Automation for BM25 and DFR Models for Model Specific optimization**

There are various sub models under DFR model and therefore there are **around 90 combinations** that have to be checked in order to judge which combination works best for our Indexed data or any data in general . This was implemented in our shell script.Similarly, in case of BM25 model, model needs to be tested for different values of k1 and b. Again, this was implemented in our shell script.

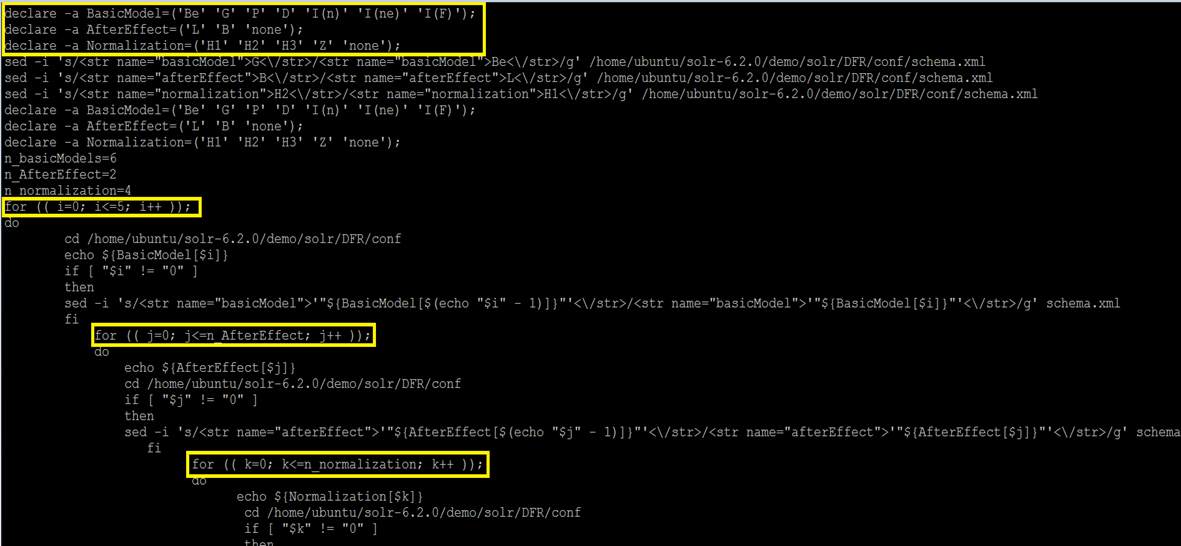


Figure 2 : Script to simulate all 90 Combinations in DFR Model

Figure Script to simulate all 441 Combinations in BM25 Model

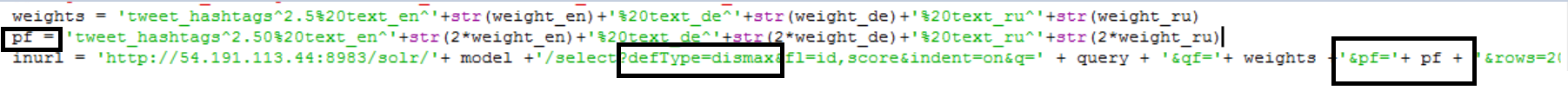
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Default Setting** | **After Language Translation** | **Query Weight Boosting** | **After**  **Synonyms** | **URL Removal** | **Model Specific Tuning** |
| VSM | 0.6469 | 0.6520 | 0.7030 | 0.7079 | 0.7271 | 0.7271 |
| DFR | 0.6468 | 0.6887 | 0.7082 | 0.7078 | 0.7132 | .7200(G, L and H1) |
| BM25 | 0.6554 | 0.6669 | 0.6873 | 0.7068 | 0.7174 | .7200(K1=1,b=.55) |

**2.9 PF (Phrase query) Set:**

**Motivation:** The phrase query helps to search documents containing query words at very close proximity such as “human rights”.

**Implementation Impact:** We tried using the phrase query to “boost” the score of the documents in cases where all of terms in the q parameter appear in close proximity.

However, in our case the value of MAP did not change after applying the phrase query. For example, phrases such as “Human rights”, “Barack Obama”, “Donald trump” did not return the appropriate results. Below is the manner in which we implemented this technique:



**2.10 Customizing default Similarity Classes :**

To achieve customization, we extended Classic Similarity class and tried various MULTIPLICATION FACTOR to original weights as returned by Class Similarity Class, namely factors **idf\_custom and tf\_custom** below.

We tried using tf\_custom and idf\_custom to improve the MAP value but it there was no increment seen.

**Java Code:**

