IR PROJECT3 REPORT

Team Members:

Tariq Siddiqui

Kaushik Ramasubramanian

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

```
<fieldType name="text_en" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <!--filter class="solr.SynonymFilterFactory" expand="true" ignoreCase="true" synonyms="synonyms.txt"/-->
    <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt" ignoreCase="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EnglishPossessiveFilterFactory"/>
    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
    <filter class="solr.PorterStemFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.SynonymFilterFactory" expand="true" ignoreCase="true" synonyms="synonyms.txt"/>
    <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt" ignoreCase="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EnglishPossessiveFilterFactory"/>
    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
    <filter class="solr.PorterStemFilterFactory"/>
  </analyzer>
```

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.

1.1 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)

```
<schema name="example-data-driven-schema" version="1.6">
  <uniqueKey>id</uniqueKey>
  <similarity class="org.apache.lucene.search.similarities.ClassicSimilarity"/>
  <!--similarity class="org.apache.lucene.search.similarities.MyNewSimilarityClass"/-->
```

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

```
  <uniqueKey>id</uniqueKey>
  <!--similarity class="org.apache.lucene.search.similarities.ClassicSimilarity"/-->
  <similarity class="org.apache.lucene.search.similarities.MyNewSimilarityClass"/>
  <fieldType name="ancestor_path" class="solr.TextField">
```

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

1.2 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:

$$\text{weight}(t|d) \propto -\log \text{Prob}_M(t \in d|\text{Collection})$$

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.

```
<schema name="example-data-driven-schema" version="1.6">
  <uniqueKey>id</uniqueKey>
  <similarity class="solr.DFRSimilarityFactory">
  <str name="basicModel">G</str>
  <str name="afterEffect">B</str>
  <str name="normalization">H2</str>
  <float name="c">1</float>
  </similarity>
```

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

1.3 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.

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

In the above equation,

tf_{td} is the frequency of term t in document d
L_d and L_{ave} are the length of document d and the average document length for the whole collection
k_1 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.

```
<uniqueKey>id</uniqueKey>
<!--similarity class="org.apache.lucene.search.similarities.BM25Similarity" /-->
<similarity class="solr.BM25SimilarityFactory">
<float name="k1">1.2</float>
<float name="b">.75</float>
</similarity>
```

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

4. Synonym Factory Implementation
5. Garbage Information(URL) Removal from Text Field
6. Exact Case matching (Using Copy Field)
7. Exact “Term” matching (Using synonym Factory and Copy Field)
8. Script Automation for BM25 and DFR Models for Model Specific optimization
9. Dismax for Phrase query
10. 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.

Original Query	:	Russia's intervention in Syria
Query translation in Russian Language	:	Вмешательство России в Сирии
Query translation in German Language	:	Russlands Intervention in Syrien

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

001 text_en:(Russia's intervention in Syria) OR text_ru:(Вмешательство России в Сирии) OR text_de:(Russlands Intervention in Syrien)

002 text_en:(US air dropped 50 tons of Ammo on Syria) OR text_ru:(США воздуха упала 50 тонн боеприпасов на Сирию) OR text_de:(US-Luft fiel 50 Tonnen Ammo auf Syrien)

003 text_en:(The European Refugee Crisis and Syria Explained animation) OR text_ru:(Европейский кризис по делам беженцев и Сирии Разъяснения анимации) OR text_de:(Die Europäische Flüchtlingskrise und Syrien Erklärten die Animation)

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

```
weights = 'tweet_hashtags^2.04%20text_en^'+str(weight_en)+'%20text_de^'+str(weight_de)+'%20text_ru^'+str(weight_ru)
pf = 'tweet_hashtags^4.0%20text_en^'+str(2*weight_en)+'%20text_de^'+str(2*weight_de)+'%20text_ru^'+str(2*weight_ru)
##print(weights)
inurl = 'http://54.191.113.44:8983/solr/'+ model +'/select?defType=dismax&fl=id,score&indent=on&q=' + query + '&qf='+ weights + '&rows=20&wt=json'
```

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.

```
with open('queries_multiple_languages.txt', encoding="utf-8") as f:
    for line in f:
        query=line[4:len(line)]
        weight_en=1.5
        weight_de=1.5
        weight_ru=1.5
        original_lang =(query[5:7])
        if original_lang=="en":
            weight_en=2.0
        elif original_lang=="de":
            weight_de=2.0
        elif original_lang=="ru":
            weight_ru=2.0

        print(original_lang)
        query = line.strip('\n').replace(':', '')
        query = urllib.parse.quote(query)

        ##weights = 'tweet_hashtags^2.04%20text_en^2.25%20text_de^2.0%20text_ru^2.0'
        weights = 'tweet_hashtags^2.04%20text_en^'+str(weight_en)+'%20text_de^'+str(weight_de)+'%20text_ru^'+str(weight_ru)
        pf = 'tweet_hashtags^4.0%20text_en^'+str(2*weight_en)+'%20text_de^'+str(2*weight_de)+'%20text_ru^'+str(2*weight_ru)
        ##print(weights)
        inurl = 'http://54.191.113.44:8983/solr/'+ model +'/select?defType=dismax&fl=id,score&indent=on&q=' + query + '&qf='+ weights + '&rows=20&wt=json'
```

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.

```
<fieldType name="text_en" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <!--filter class="solr.SynonymFilterFactory" expand="true" ignoreCase="true" synonyms="synonyms.txt"/
    <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt" ignoreCase="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EnglishPossessiveFilterFactory"/>
    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
    <filter class="solr.PorterStemFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.SynonymFilterFactory" expand="true" ignoreCase="true" synonyms="synonyms.txt"/>
    <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt" ignoreCase="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EnglishPossessiveFilterFactory"/>
    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
    <filter class="solr.PorterStemFilterFactory"/>
  </analyzer>
</fieldType>
```

```
# Some synonym groups specific to this example
#GB,gib,gigabyte,gigabytes
#MB,mib,megabyte,megabytes
#Television, Televisions, TV, TVs
#notice we use "gib" instead of "GiB" so any WordDelimiterFilter coming
#after us won't split it into two words.
##launch,host
u.s.,U.S.,USA,usa,США,washington
isis,isil
russia,russland
russischen,russian
syrian,syrischen,syrien
terrorist,rebel,militant
ammo,ammunition,munition
say,talk
animated,animation
officer,general,commander
aid,fund,Aid,fonds,relief,philanthropy
dead,deceased
killed,shot
beat,slam
march,rally
vladimir,putin
barack,obama
shells,rockets,missile
hit,attack,strike,raid
bomb,airdrop,airstrike,air drop
jet,airforce,aircraft,air force
poll,survey
support,campaign
штурмуют,подъеме
techcrunch,airbnb,instacart,kickstarter,tech company,startup,firm
облегчение,помощь,фонд
```

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.

```
import re
import json

without_hashtag = []

url_regex = r'(https|http)?:\/\/(\w|\.|\/|\?|\=|\&|\%)*\b'
pattern = re.compile(url_regex)

with open('train.json',encoding="utf-8") as f:
        tweet = json.load(f)

        for value in tweet:
                if len(value["text_de"]) != 0:
                        value["text_de"] = pattern.sub('', value["text_de"])

                elif len(value["text_ru"]) != 0:
                        value["text_ru"] = pattern.sub('', value["text_ru"])

                elif len(value["text_en"]) != 0:
                        value["text_en"] = pattern.sub('', value["text_en"])

                without_hashtag.append(value)

outputFile = open('new_train.json','w')
outputFile.write(json.dumps(without_hashtag,ensure_ascii=False))
```

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.

```
<field name="lang" type="strings"/>
<field name="text_de" type="text_de"/>
<field name="text_en" type="text_en"/>
<field name="text_ru" type="text_ru"/>
<field name="text_de_Without_lowercasing" type="text_de_Without_lowercasing"/>
<field name="text_en_Without_lowercasing" type="text_en_Without_lowercasing"/>
<field name="text_ru_Without_lowercasing" type="text_ru_Without_lowercasing"/>
```

```
weights = 'text_en_no_lower_casing^3.0%20text_en^2.0%20text_de_no_lower_casing^3.0%20text_de^2.0%20text_ru_no_lower_casing^3.0%20text_ru^2.0
inurl = 'http://54.191.113.44:8983/solr/'+ model +'/select?fl=id,score&indent=on&q=' + query + '&rows=20&wt=json'
```

```
<fieldType name="text_en_Without_lowercasing" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt" ignoreCase="true"/>
    <!--filter class="solr.LowerCaseFilterFactory"/-->
    <filter class="solr.EnglishPossessiveFilterFactory"/>
    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
    <filter class="solr.PorterStemFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.SynonymFilterFactory" expand="true" ignoreCase="true" synonyms="synonyms.txt"/>
    <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt" ignoreCase="true"/>
    <!--filter class="solr.LowerCaseFilterFactory"/-->
    <filter class="solr.EnglishPossessiveFilterFactory"/>
    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
    <filter class="solr.PorterStemFilterFactory"/>
  </analyzer>
</fieldType>
```

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.

```
<field name="lang" type="strings"/>
<field name="text_de" type="text_de"/>
<field name="text_en" type="text_en"/>
<field name="text_ru" type="text_ru"/>
<field name="text_de_Without_Synonym" type="text_de_Without_Synonym"/>
<field name="text_en_Without_Synonym" type="text_en_Without_Synonym"/>
<field name="text_ru_Without_Synonym" type="text_ru_Without_Synonym"/>
```

```
<fieldType name="text_en_Without_Synonym" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt" ignoreCase="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EnglishPossessiveFilterFactory"/>
    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
    <filter class="solr.PorterStemFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <!--filter class="solr.SynonymFilterFactory" expand="true" ignoreCase="true" synonyms="synonyms.txt"/-->
    <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt" ignoreCase="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EnglishPossessiveFilterFactory"/>
    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
    <filter class="solr.PorterStemFilterFactory"/>
  </analyzer>
</fieldType>
```

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.

```
declare -a BasicModel=('Be' 'G' 'P' 'D' 'I(n)' 'I(ne)' 'I(F)');
declare -a AfterEffect=('L' 'B' 'none');
declare -a Normalization=('H1' 'H2' 'H3' 'Z' 'none');
sed -i 's/<str name="basicModel">G<\/str>/<str name="basicModel">Be<\/str>/g' /home/ubuntu/solr-6.2.0/demo/solr/DFR/conf/schema.xml
sed -i 's/<str name="afterEffect">B<\/str>/<str name="afterEffect">L<\/str>/g' /home/ubuntu/solr-6.2.0/demo/solr/DFR/conf/schema.xml
sed -i 's/<str name="normalization">H2<\/str>/<str name="normalization">H1<\/str>/g' /home/ubuntu/solr-6.2.0/demo/solr/DFR/conf/schema.xml
declare -a BasicModel=('Be' 'G' 'P' 'D' 'I(n)' 'I(ne)' 'I(F)');
declare -a AfterEffect=('L' 'B' 'none');
declare -a Normalization=('H1' 'H2' 'H3' 'Z' 'none');
n_basicModels=6
n_AfterEffect=2
n_normalization=4
for (( i=0; i<=5; i++ ));
do
        cd /home/ubuntu/solr-6.2.0/demo/solr/DFR/conf
        echo ${BasicModel[$i]}
        if [ "$i" != "0" ]
        then
        sed -i 's/<str name="basicModel">'"${BasicModel[$(echo "$i" - 1)]}"'<\/str>/<str name="basicModel">'"${BasicModel[$i]}"'<\/str>/g' schema.xml
        fi
            for (( j=0; j<=n_AfterEffect; j++ ));
            do
                echo ${AfterEffect[$j]}
                cd /home/ubuntu/solr-6.2.0/demo/solr/DFR/conf
                if [ "$j" != "0" ]
                then
                sed -i 's/<str name="afterEffect">'"${AfterEffect[$(echo "$j" - 1)]}"'<\/str>/<str name="afterEffect">'"${AfterEffect[$j]}"'<\/str>/g' schema
                   fi
                        for (( k=0; k<=n_normalization; k++ ));
                        do
                              echo ${Normalization[$k]}
                               cd /home/ubuntu/solr-6.2.0/demo/solr/DFR/conf
                               if [ "$k" != "0" ]
                               then
```

Figure 22 : Script to simulate all 90 Combinations in DFR Model

```
rm BM25_Combinations.txt
cd /home/ubuntu/solr-6.2.0/demo/solr/BM25/conf
rm schema-Orig.xml
cp schema.xml schema-Orig.xml
sed -i 's/<float name="k1">1<\/float>/<float name="k1">0.95<\/float>/g' schema.xmil
sed -i 's/<float name="b">1<\/float>/<float name="b">-0.05<\/float>/g' schema.xml
current_k1=0.95
current_b=-0.05
offset=0.05
for (( i=0; i<=20; i++ ));
do
        current_b=-0.05
        cd /home/ubuntu/solr-6.2.0/demo/solr/BM25/conf
        to_update_k1=$(echo "$current_k1" + "$offset"|bc)
        echo $to_update_k1
        sed -i 's/<float name="k1">'"${current_k1}"'<\/float>/<float name="k1">'"${to_update_k1}"'<\/float>/g' schema.xml
        current_k1=$to_update_k1
            for (( j=0; j<=20; j++ ));
            do
                to_update_b=$(echo "$current_b" + "$offset"|bc)
                cd /home/ubuntu/solr-6.2.0/demo/solr/BM25/conf
                #echo "inside loop"
                sed -i 's/<float name="b">'"${current_b}"'<\/float>/<float name="b">'"${to_update_b}"'<\/float>/g' schema.xml
                current_b=$to_update_b
                cd /home/ubuntu/auntomation/BM25
                echo "New Combination"  >> BM25_Combinations.txt
                echo $current_k1 >> BM25_Combinations.txt
                echo $current_b >> BM25_Combinations.txt
                sh ./script_BM25
            done
                cd /home/ubuntu/solr-6.2.0/demo/solr/BM25/conf
                sed -i 's/<float name="b">'"${current_b}"'<\/float>/<float name="b">-0.05<\/float>/g' schema.xml
done
```

Figure 3Script 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:

```
weights = 'tweet_hashtags^2.5%20text_en^'+str(weight_en)+'%20text_de^'+str(weight_de)+'%20text_ru^'+str(weight_ru)
pf = 'tweet_hashtags^2.50%20text_en^'+str(2*weight_en)+'%20text_de^'+str(2*weight_de)+'%20text_ru^'+str(2*weight_ru)
inurl = 'http://54.191.113.44:8983/solr/'+ model +'/select?defType=dismax&fl=id,score&indent=on&q=' + query + '&qf='+ wei
```

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:

```
package org.apache.lucene.search.similarities;
import org.apache.lucene.search.similarities.ClassicSimilarity;;

public class MyNewSimilarityClass extends ClassicSimilarity {


@Override
public float idf(long docFreq, long numDocs) {
    float idf_custom =1;
    return (idf_custom*super.idf(docFreq, numDocs));
}

@Override
public float tf(float freq) {
    float tf_custom =1;
    return (tf_custom*super.tf(freq));
}
}
```