Information Retrieval Project 3

Evaluation of IR Models

Team#: 26
Saurabh Bajoria - 50208005
Sumedh Ambokar- 50207865

Introduction:

Calculating Mean Average Precision for a given set of queries based on a given set of tweets for implementation of **Vector Space Model**, **Divergence-from-randomness** and **Best Matching 25 (BM25).**

Solr implements the Similarity Models using the Similarity Factory classes declared by Apache **Lucene**. Appropriate implementation of a Similarity Model improves the ranking and scoring of the documents while indexing and querying. The ranked documents have to be compared with a predefined ranking of documents based on the manual relevance calculation using **trec_eval** comparing application. Final output of the MAP defines efficiency of the Model designed.

Similarity Models

Similarity Model can be defined in **schema.xml** of a Solr core. Each core collection has one "global" Similarity, and by default Solr uses an implicit SchemaSimilarityFactory which allows individual field types to be configured with a "per-type" specific Similarity and implicitly uses BM25Similarity (BM25 Model) for any field type which does not have an explicit Similarity. This default behavior can be overridden by declaring a top-level element in your schema.xml, outside of any single field type. Such similarity declaration can either refer directly to the name of a class with a no-argument constructor, such as the below example showing BM25Similarity:

```
<similarity class=”solr.BM25Similarity”/>
```

or by referencing a SimilarityFactory implementation, which may take optional initialization parameters:

```
<similarity class="solr.DFRSimilarityFactory">
<str name="basicModel">Be</str>
<str name="afterEffect">L</str>
<str name="normalization">H1</str>
</similarity>
```

Implementation of Default Models

We implemented three models with their default configurations and extracted scoring for all the queries provided.

Vector Space Model

Class: *solr.ClassicSimilarityFactory*

Parameters: None

Screenshot of the implementation:

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

BM25: It is based on the probabilistic retrieval framework.

Class: *solr.BM25SimilarityFactory*

Parameters: k1, b

Default Values: k1=1.2 b=0.75

Range of Values: k1 € [1.2, 2] b € [0.5, 0.8]

Screenshot of the implementation:

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

Divergence from Randomness: It is based on the idea that the term-weight is inversely related to the probability of term-frequency within the document

Class: solr.DFRSimilarityFactory

Parameters: basicModel, afterEffect, normalization

Default Values: basicModel = G, aftereffect= B, normalization=H2

Range of Values: basicModel € {Be, G, P, D, I(n), I(ne), I(F)}
afterEffect € {L, B, none}
normalization € {H1, H2, H3, Z, none}

Screenshot of the implementation:

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

Map Values using Default Implementation:

Similarity Models used	MAP value
BM25 (BM25SimilarityFactory)	**0.6689**
DFR (DFRSimilarityFactory)	**0.6712**
VSM (ClassicSimilarityFactory)	**0.6617**

Considering these MAP values from above table as a reference we need to try and better the MAP values with appropriate implementation of the each Similarity model and improvements if possible.

Attempted Improvements:

1. Query in multilingual text fields:

Assumption:

On analyzing all the queries and their subsequent outputs, it was observed that restricting a query to a particular language text field might hamper the value of the final Map as it affects the recall and precision values for each query.

Adhering to the above assumption we tried to run every query for all the text fields i.e. “text_en”, “text_ru” and “text_de” by changing value of “df” parameter.

Implementation:

To get the required output a python script was used which would iterate on every text field and change the value of “df” field while querying on the solr indexed file. Python script retrieves the Json file from solr using following http request:

```
with open('queries.txt', encoding="utf-8") as f:
    for line in f:
        query = line.strip('s\n').replace(':', '')[4:]
        query = urllib.parse.quote(query)
        countRows = 1
        for lang in langs:
            inurl = ('http://ec2-35-163-27-180.us-west-2.compute.amazonaws.com:8983/solr/' + core +
                '/select?df=' + lang + '&fl=score,id&indent=on&q=' + query + '&rows=20&wt=json')
```

Here lang is a list of strings storing the related text fields.

Output:

Similarity Models used	MAP value
BM25 (BM25SimilarityFactory)	**0.6815**
DFR (DFRSimilarityFactory)	**0.6888**
VSM (ClassicSimilarityFactory)	**0.6831**

Conclusion:

From the table we can conclude that using the query for all the fields has increased the MAP value significantly.

For all the further assumptions query was run for all the language text fields as MAP value has a significantly better output.

2. Modifying the Similarity Factory parameters:

Assumption:

Default parameters provided by solr for the similarity classes might be hampering better scoring for some queries. Thus we need to evaluate all the combinations of the parameters passed to the Similarity Class i.e. SimilarityFactory classes.

Implementation:

We need to modify the schema file for all the cores which are representation of individual modules i.e. BM25, DFR and VSM (Classic) and add the combination of the respective parameters.

For this implementation a Java class was created which performed following processes:

- Modified the schema dynamically
- Restarted the solr core
- Re-posted the json data file
- Run all the queries on the solr
- Print the output values in a text file
- Run trec_eval command for every file created
- Get the MAP values for every output

The implementation is restricted to modules BM25 and DFR as only these similarity modules support changes in parameters to the Similarity factory class.

Output:

Best values obtained from the above implementation are:

Similarity Models used	MAP value	Parameter Values
BM25 (BM25SimilarityFactory)	**0.6924**	**k1 = 1.3 | b = 0.76**
DFR (DFRSimilarityFactory)	**0.6888**	**basicModel = P | afterEffect = L | normalization = H2**

Conclusion:

Better results of MAP were obtained for both the modules on changing the values of the parameters as per the given table above.

3. StopWord Filter

Assumption

Adding StopWord Filter increases recall and reduces precision, thus affecting the overall MAP value.

Implementation

StopWord Filter was added in the schema.xml for all the language specific fields, using the below:

```
<filter class="solr.StopFilterFactory"
        ignoreCase="true"
        words="lang/stopwords_en.txt"
        />
```

Stop words are included in the language specific text file, i.e. stopwords_en.txt for English and so on.

Output

Classic		BM25		DFR	
Without	With	Without	With	Without	With
0.6824	0.6831	0.6812	0.6815	0.6886	0.6888

*With – with StopFilterFactory

*Without – without StopFilterFactory

Conclusion

It is observed from the above table that the MAP value has marginally improved for BM25 and DFR models. This change is not significant as number of query rows are fixed, i.e. 20 for every query.

But for Classic model, a significant increase has been observed.

4. Query Expansion

Assumption

Adding Synonym Filter Factory increases recall, thus affecting the overall MAP value

For each term *t*, the query can be expanded with synonyms and related words of *t* from the thesaurus.

Implementation

Synonym Filter is added to the schema.xml using the below:

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

Synonyms of words are added in the synonyms.txt file

Output

Classic		BM25		DFR	
Without	With	Without	With	Without	With
0.6831	0.6848	0.6815	0.6831	0.6888	0.6892

*Without – without SynonymFilterFactory

*With – with SynonymFilterFactory

Conclusion

It is observed from the above table that the MAP values have increased considerably for all the models when synonyms are added.

5. Language translation querying

Assumption

Queries are language specific. Restricting the queries in just one language might hamper MAP values. Thus every query can be translated to their language counterparts. Such translated queries when used with the language specific text fields will ideally help in improving resulting MAP values.

Implementation

All the queries are converted to all the languages and each query is executed for every specific language text field. Queries were translated manually and executed using python script to get the final MAP values for each set of translated queries.

Output

Similarity Models used	MAP value
BM25 (BM25SimilarityFactory)	**0.6754**
DFR (DFRSimilarityFactory)	**0.7012**
VSM (ClassicSimilarityFactory)	**0.6795**

Conclusion

Translation of the queries has yield better values of MAP. Translation has increased the retrieval of relevant documents at higher rank thus provided a better MAP.

6. Stem Filter

Assumption:

Stem filters are used for obtaining the root words of the queried words while indexing and querying. Stemming helps in improving recall. Use of stemmer will help in obtaining more relevant documents from the set of the documents. Stemming is specific to language based on the characteristics of the individual language.

Implementation:

Solr implements stemming by using FilterFactory classes from Apache Lucene. Filter needs to be added to the schema for language specific text field. Below is the list of filter used:

Field	Filter Used
text_en	PorterStemFilterFactory
text_ru	SnowballPorterFilterFactory
text_de	GermanLightStemFilterFactory

MAP values were tested for schemas with the filters and without filters.

Output:

Classic		BM25		DFR	
Without	With	Without	With	Without	With
0.6052	0.6831	0.6245	0.6815	0.6269	0.6888

*With – with Stem Filters

*Without – without Stem Filters

Conclusion:

Referring to the table we can conclude that inclusion of the stemming filters increases the MAP values significantly.

7. Field Boost

Assumption:

Field boost helps to assign a particular field as a high preference field. Any document matching such criteria will be marked at a higher score and thus will have a higher rank. Such boosts can assist in improving precision for certain queries e.g. Queries with hashtags can be queried using a boost field **hashtags** in solr.

Implementation:

In solr a field can be marked as a Boost Field by assigning that field as the default field i.e. using **df** field while querying in solr.

Conclusion:

Such implementation will be helping to improve query specific document scoring values. Thus this improvement will be query specific and cannot be implemented for a generic set of queries. Therefore, this improvement technique is not used in our implementation of Similarity models.

8. Query Boost

Assumption:

By using Query-time boosts, we assign higher score to a document based on an occurrence of a particular word or words. As query boosting improves scoring there are high chances of more relevant documents having a high score, thus improving MAP score.

Implementation:

In Solr, any word in the query can be boosted using the 'bq' field under edismax. Also, the weight by which the word needs to be boosted can be specified.

Conclusion:

Query boosting helps improve scoring of certain specific queries. But it cannot be applied to the overall system, thus this improvement technique is not implemented.

Final Conclusion:

Based on the improvement techniques used and the conclusions derived from their outputs we can design a final implementation for every Similarity Model. The MAP values have been fairly stable due to restricted number of returned rows from every query, i.e 20. Precision and recall both are affected because of this restriction.

Below are the changes implemented for all the models:

- Stop Word Filter added
- Stemming Filter added
- Synonym Filter added

Below are the changes implement for specific model:

[Parameters for the below Models have been identified after modifying Similarity Factory parameters for every Model]

1. **BM25 Model**

- Parameters**:**

Parameter Name	Value
k1	**1.3**
B	**0.76**

- Query in multilingual text fields has been implemented.

2. **DFR Model**

- Parameters:

Parameter Name	Value
basicModel	**P**
afterEffect	**L**
normalization	**H2**

- Language translation querying has been implemented.

3. **VSM Model**

- Language translation querying has been implemented.

References:

http://lucene.apache.org/solr/4_0_0/solr-core/org/apache/solr/search/similarities/package-summary.html

https://lucene.apache.org/core/2_9_4/api/all/org/apache/lucene/search/Similarity.html

https://wiki.apache.org/solr/SolrRelevancyFAQ#How_can_I_make_exact-case_matches_score_higher