CSE 535 Information Retrieval

Project 3: Evaluation of IR Models

Namrata Bakre
University at Buffalo,
Buffalo, NY
UB Person ID: 50336849
Email: nbakre@buffalo.edu
UB Person Name: nbakre

***Abstract*—** The goal of this project is to implement various IR models, evaluate the IR system and improve the search result based on the understanding of the models, the implementation and the evaluation. The idea behind the project is with given Twitter data in three languages - English, German and Russian, 15 sample queries and the corresponding relevance judgement, index that data by Solr, implement Vector Space Model and BM25 based on Solr, and evaluate the two sets of results using Trec Eval program. Based on the evaluation result, improve the performance in terms of the measure Mean Average Precision (MAP).

I. INTRODUCTION

Information Retrieval (IR) can be defined as a software program that deals with the organization, storage, retrieval, and evaluation of information from document repositories, particularly textual information. Information Retrieval is the activity of obtaining material that can usually be documented on an unstructured nature i.e. usually text which satisfies an information need from within large collections which is stored on computers. For example, Information Retrieval can be when a user enters a query into the system.

In this project, after indexing the given twitter data by Solr and implementing Vector Space Model and BM25 based on Solr, for the evaluation of two sets of results we have used Trec_Eval(Text REtrieval Conference) program. trec_eval is the standard tool used by the TREC community for evaluating an ad hoc retrieval run, given the results file and a standard set of judged results (relevance judgements). The input file format for TREC is: query-number Q0 tweet_id rank similarity_score model_name

The dataset used for this project is Twitter data saved in json format - train.json. There are three languages included in the data: English (text_en), German (text_de) and Russian (text_ru). There are 15 sample queries (queries.txt) and corresponding manually judged relevance score (qrel.txt) provided with this data. The IR model is tested against these 15 queries.

Task Definition: Given Twitter data, sample queries and the corresponding relevance judgement score for each query, do the following:

1. Implement and evaluate Vector Space Model and BM25 on Solr

2. Based on evaluation results, improve performance of these 2 models in terms of MAP (Mean Average Precision) score.

II. IMPLEMENTATION

The first step in the implementation part is to index data against two models – BM25 and VSM.

A) Best Matching (BM25) Model:

Each 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 for any field type which does not have an explicit Similarity. We have implemented BM25 Model in solr by creating a BM25 Model core and using default schema file.

BM25 Model has 2 parameters k1 and b. k1 controls non-linear term frequency normalization (saturation) and its default value is 1.2. b parameter controls to what degree document length normalizes tf values and its default value is 0.75. A higher k1 means that the score for each term can continue to go up by relatively more for more instances of that term. A value of 0 for k1 would mean that everything except IDF(qi) would cancel out. Following table shows slight increment in MAP values with different pairs of k1 and b values.

	Default	Modifications
k1	**1.2**	**0**
b	**0.75**	**0.3**
MAP Value	**0.2497**	**0.2739**

B) Vector Space Model (VSM):

The representation of a set of documents as vectors in a common vector space is known as the vector space model and is fundamental to a host of information retrieval operations ranging from scoring documents on a query, document classification and document clustering.

Implementation of VSM model is done by modifying the schema file as below:

<similarity class="solr.ClassicSimilarityFactory"/>

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Solr managed schema - automatically generated - DO NOT EDIT -->
<schema name="default-config" version="1.6">
  <uniqueKey>id</uniqueKey>
  <similarity class="solr.ClassicSimilarityFactory"/>
  <fieldType name="_nest_path_" class="solr.NestPathField" maxCharsForDocValues="
  <fieldType name="ancestor_path" class="solr.TextField">
```

III. PLAN

In this section, we discuss measures taken to improve the IR system. Together with the training queries, query results, ground truth judgements and the TREC_eval result, we have gained an intuition on the performance of our IR system. Users always want relevant documents retrieved for the query they run. Giving higher relevance to a set of documents over others is called boosting.

We choose the measure MAP as main objective to improve.

(i) Boosting with eDismax Query Parser:

The Extended DisMax (eDisMax) query parser is an improved version of the 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. Additional options enable users to influence the score based on rules specific to each use case (independent of user input).

We got below results with Solr's default query parser known as the 'lucene' parser. Here total number of documents returned are only **13.**

The MAP value obtained after parsing all the test queries with solr's default parser was **0.2497**

We have made changes in solrconfig.xml to use eDismax query parser.

solrconfig.xml changes:

```
<requestHandler name="/select" class="solr.SearchHandler">
  <!-- default values for query parameters can be specified, these
       will be overridden by parameters in the request
  -->
  <lst name="defaults">
    <str name="echoParams">explicit</str>
    <int name="rows">10</int>
      <str name="defType">edismax</str>
    <str name="qf">text_en text_de text_ru</str>
    <!-- Default search field
      <str name="df">text</str>
    -->
    <!-- Change from JSON to XML format (the default prior to Solr 7.0)
      <str name="wt">xml</str>
    -->
  </lst>
```

After parsing the queries with eDismax query parser, the total number of documents returned significantly increased **from 13 to 939.**

Also, there was a significant improvement in the MAP values after parsing the queries with eDismax query parser.

The MAP value got after parsing the queries with eDismax query parser is **0.6850**

For BM25:

	MAP
Default Query Parser	**0.2497**
eDismax Query Parser	**0.6850**

For VSM:

	MAP
Default Query Parser	**0.2430**
eDismax Query Parser	**0.6771**

(ii) Using various Filter Factories:

Filters consume one TokenStream and produce a new TokenStream, they can be chained one after another indefinitely. Each filter in the chain in turn processes the tokens produced by its predecessor. The order in which you specify the filters is therefore significant. Typically, the most general filtering is done first, and later filtering stages are more specialized.

Solr includes several language-specific stemmers created by the Snowball generator that are based on the Porter stemming algorithm. The generic Snowball Porter Stemmer Filter can be used to configure any of these language stemmers.

For English:

```
<fieldType name="text_en" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.SynonymGraphFilterFactory" 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>
```

For German:

```
<fieldType name="text_de" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" format="snowball" words="lang/stopwords_de.txt" ignoreCase="true"/>
    <filter class="solr.GermanNormalizationFilterFactory"/>
    <filter class="solr.GermanLightStemFilterFactory"/>
  </analyzer>
```

For Russian:

```
<fieldType name="text_ru" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" format="snowball" words="lang/stopwords_ru.txt" ignoreCase="true"/>
    <filter class="solr.SnowballPorterFilterFactory" language="Russian"/>
  </analyzer>
</fieldType>
```

IV. RESULTS

Following chart shows the MAP values for Improved IR Model and what measures are consider for the improvement:

	Default MAP Values	**After tweaking parameters**	**After using eDismax query parser**	**Final MAP Values**
BM25	0.2497	0.2739	0.6850	0.6850
VSM	0.2430	0.2430	0.6771	0.6771

Final MAP Value for BM25:

```
_
P_1000                  015 0.0130
runid                   all bm25
num_q                   all 15
num_ret                 all 280
num_rel                 all 225
num_rel_ret             all 119
map                     all 0.6850
gm_map                  all 0.6113
Rprec                   all 0.6729
bpref                   all 0.6879
recip_rank              all 1.0000
iprec_at_recall_0.00    all 1.0000
iprec_at_recall_0.10    all 0.9762
iprec_at_recall_0.20    all 0.9333
iprec_at_recall_0.30    all 0.9067
```

Final MAP Value for VSM:

```
_
P_1000                  015 0.0130
runid                   all vsm
num_q                   all 15
num_ret                 all 280
num_rel                 all 225
num_rel_ret             all 120
map                     all 0.6771
gm_map                  all 0.6064
Rprec                   all 0.6498
bpref                   all 0.6895
recip_rank              all 1.0000
iprec_at_recall_0.00    all 1.0000
iprec_at_recall_0.10    all 1.0000
iprec_at_recall_0.20    all 0.9259
iprec_at_recall_0.30    all 0.8847
```

V. CONCLUSION

Hence, we have successfully implemented the BM25 Model and VSM Model and obtained optimal MAP values for each of them by their TREC evaluation.