



University at Buffalo
The State University of New York

INTRODUCTION TO INFORMATION RETRIEVAL

CSE 535

FALL-2016

PROJECT-3 REPORT

EVALUATION OF IR MODELS

SUBMITTED BY:

HEMA MADHAV JAKKAMPUDI(50206563)(hemamadh)

ARUN CHANDRA PENDYALA(50207136)(apendyal)

TEAM NUMBER - 80

EVALUATION OF INFORMATION RETRIEVAL MODELS

1. INTRODUCTION

This project deals with the implementation of IR models such as Vector Space Model, BM25 model, DFR models based on Solr using twitter data and the results are evaluated using TREC_eval program.

We are given 20 training queries and 10 testing queries in languages – German , English and Russian. The key concept of this project is to improve the performance of IR system by considering primarily MAP(mean average precision)as an evaluation measure.

2. OVERALL METHODOLOGY

Since the intention of this project is to improve the MAP (mean average precision) value , we have experimented the three models – VSM, BM25 and DFR under different set of circumstances and the mean average precision values for each of these are collected. Initially, the MAP values are determined for the standard query parser for the three similarities. Then these MAP values are compared with implementations using query parsers like dismax and edismax query parsers. To improve the MAP value further, a custom query parser has been constructed extending from edismax query parser plugin.

We have also collected the MAP values by introducing synonyms list to each model after the careful examination of the queries.

3. EXPERIMENTATION:

3.1 DEFAULT SETUP:

We created 3 cores – one for each model – VSM(Vector Space Model) , BM25, DFR(Divergence from randomness) by modifying the schema.xml of each model. The following similarity classes have been used for each model:

- i) VSM
In Vector Space Model(VSM), the queries and documents are represented as vectored quantities in multi-dimensional space where each index item is a dimension and weights are TF-IDF values. In Solr, the corresponding similarity class is solr.ClassicSimilarity
- ii) BM25 –
Okapi BM25 model is a probabilistic information retrieval model which was originally designed for short-length documents. In Solr, the similarity class for this is solr.BM25SimilarityFactory
- iii) DFR
In Divergence from randomness model, the term-weight is inversely related to the probability of term-frequency within the document obtained by a model of randomness. In Solr, the similarity class for this is given by solr.DFRSimilarityFactory.

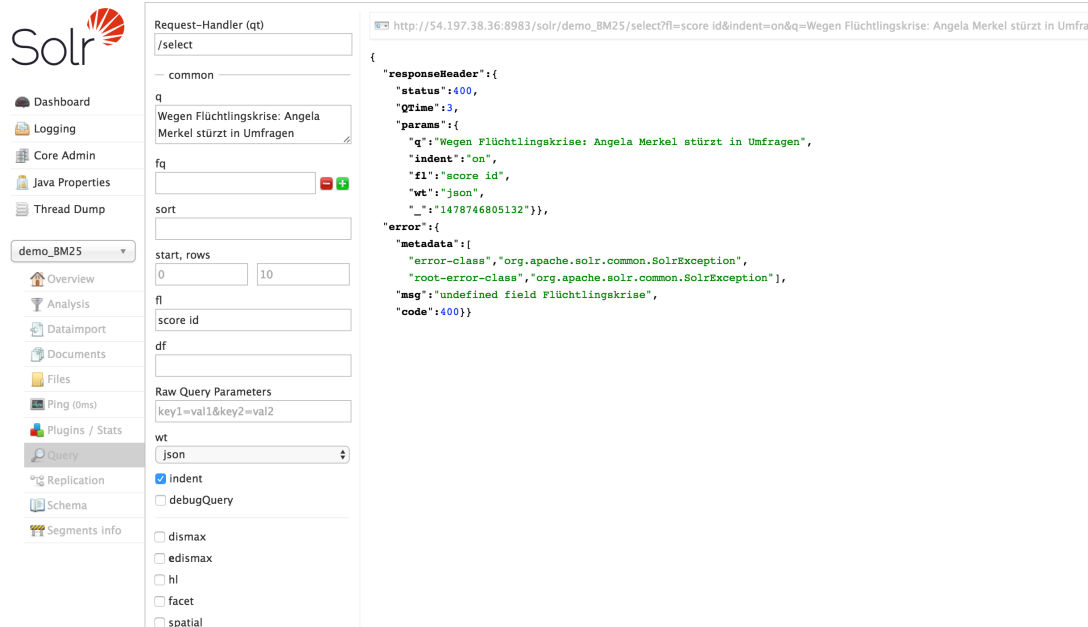
Using the default settings which has standard query parser, we obtained quite low values. The MAP values were found out to be as follows for the default settings:

- i) VSM - 0.6418
- ii) BM25 – 0.6575 - default $k_1 = 1.2$ and $b = 0.75$
- iii) DFR – 0.6554 – given defaults – H2 normalization, Basic model G and Bernoulli

We developed a python script which parses the queries text file one by one and returns the query results into a new text file for each of the models by running the query url such that 3 text files are produced corresponding to each set of the core(demo_VSM, demo_BM25, demo_DFR) and query list. Another Python script was also developed which runs the TREC_eval program for each text file corresponding to each model.

Initially we queried against the given sets of training and testing queries. We observed the tweets which were returned as results and we noticed the changes in results returned and impact on scores by making minor changes to the queries. This was much more noticeable in testing queries. For example, U.S.A can also be mapped into U.S , U S and United States of America. So we decided to create a list of synonyms based on the queries given in the training and testing sets. This is a technique of query expansion which has been discussed later in this report. We also tested out the case sensitivity of the queries and as expected, there were no changes in the results as the queries are changed to lowercase at query time and documents are set to lowercase at indexing time. Presence of special symbols such as #, @, - , : etc contribute to changes in the returned results score and their relative ranking. In some cases, no results were returned at all. So query parsing plays a crucial role in the improvement of system performance.

In order to enhance the MAP, the settings and parameters of the models are modified. The standard query parser, which is the default query parser, is intolerant of syntax errors as it expects the query to be well-formed. On the other hand, DisMax query parser is known to be a more forgiving parser as it is useful for directly passing in a user-supplied query string.



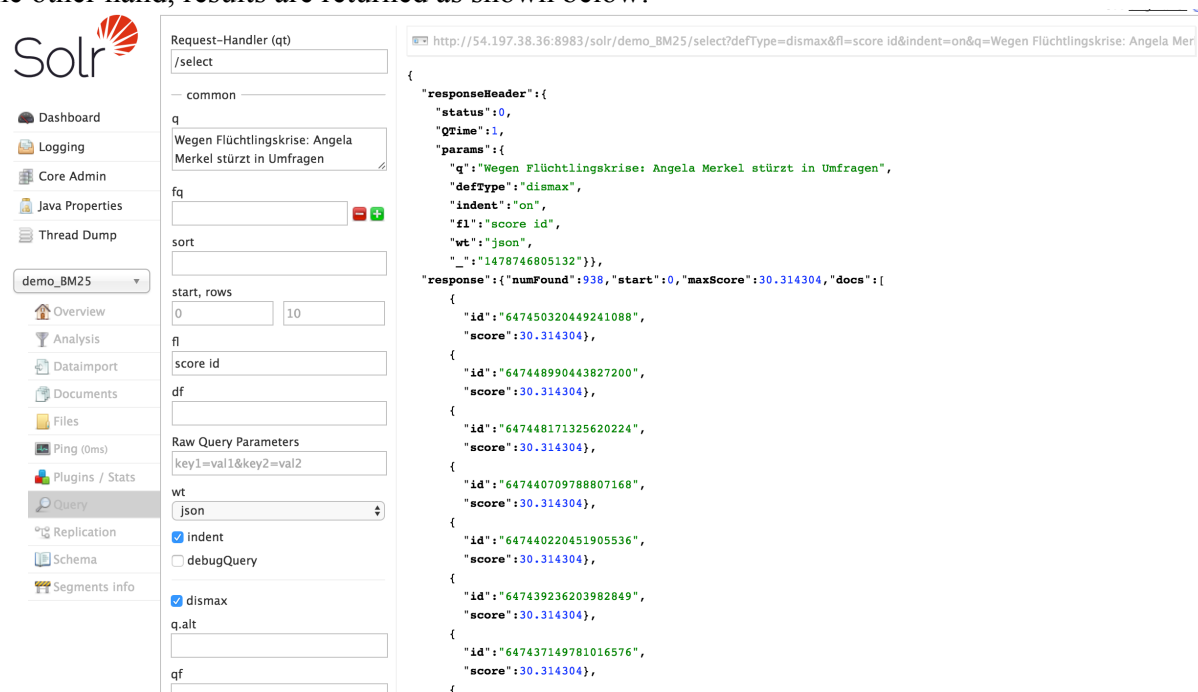
The screenshot shows the Solr Admin UI with the 'Query' tab selected. The request handler is '/select'. The query is 'Wegen Flüchtlingskrise: Angela Merkel stürzt in Umfragen'. The response is a JSON object indicating an error: 'undefined field Flüchtlingskrise' with a status of 400.

```

{
  "responseHeader": {
    "status": 400,
    "QTime": 3,
    "params": {
      "q": "Wegen Flüchtlingskrise: Angela Merkel stürzt in Umfragen",
      "indent": "on",
      "fl": "score id",
      "wt": "json",
      "_": "1478746805132"
    },
    "error": {
      "metadata": {
        "error-class": "org.apache.solr.common.SolrException",
        "root-error-class": "org.apache.solr.common.SolrException",
        "msg": "undefined field Flüchtlingskrise",
        "code": 400
      }
    }
  }
}

```

For example, the query 4 does not return any results when the standard query parser is used. On the other hand, results are returned as shown below:



The screenshot shows the Solr Admin UI with the 'Query' tab selected. The request handler is '/select'. The query is 'Wegen Flüchtlingskrise: Angela Merkel stürzt in Umfragen'. The response is a JSON object showing search results using the DisMax parser. The 'numFound' is 938, and the 'docs' array contains 6 document entries with their IDs and scores.

```

{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": {
      "q": "Wegen Flüchtlingskrise: Angela Merkel stürzt in Umfragen",
      "defType": "dismax",
      "indent": "on",
      "fl": "score id",
      "wt": "json",
      "_": "1478746805132"
    },
    "response": {
      "numFound": 938,
      "start": 0,
      "maxScore": 30.314304,
      "docs": [
        {
          "id": "647450320449241088",
          "score": 30.314304
        },
        {
          "id": "647448990443827200",
          "score": 30.314304
        },
        {
          "id": "647448171325620224",
          "score": 30.314304
        },
        {
          "id": "647440709788807168",
          "score": 30.314304
        },
        {
          "id": "647440220451905536",
          "score": 30.314304
        },
        {
          "id": "647439236203982849",
          "score": 30.314304
        },
        {
          "id": "647437149781016576",
          "score": 30.314304
        }
      ]
    }
  }
}

```

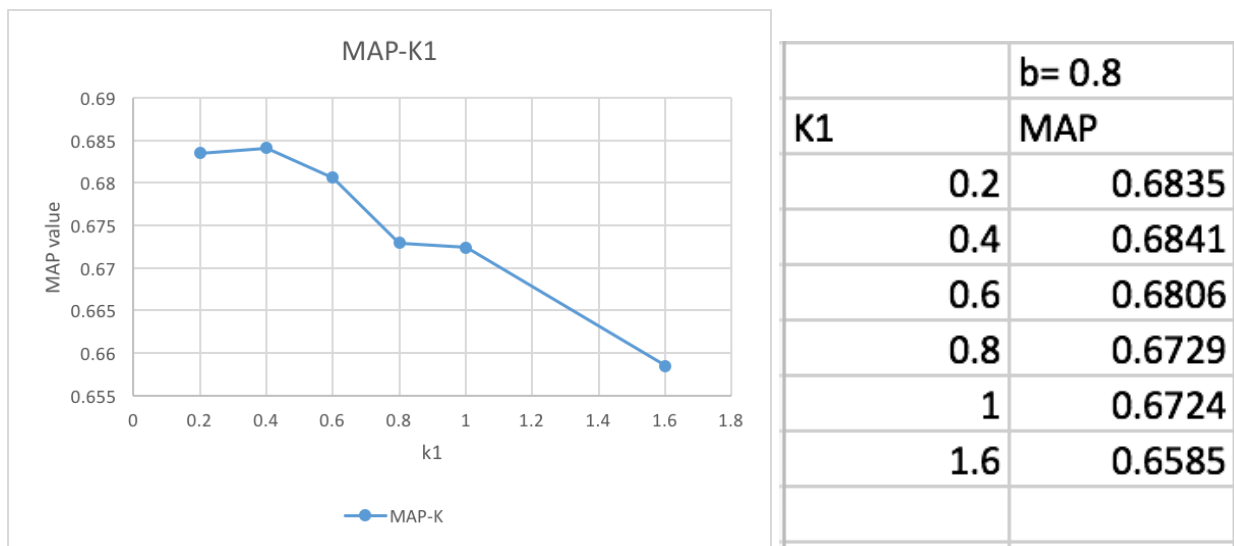
The MAP values of the models were same as the previous case with standard query parser when we used training queries. On this basis, we have chosen the DisMax query parser instead of the standard query parser.

3.3. Tuning the values of parameters for models

The tuning process involves the tweaking of parameters in order to improve the performance of the models.

3.2.1. Tuning B and K1 values in BM25:

We tried out tuning the parameters of BM25 model which has the default setting: $b = 0.75$ and $k1 = 1.2$. For the default setting, $MAP = 0.67$. Initially we varied the values of b keeping $k1 = 1.2$ and we observed that the MAP value remained unchanged at $MAP = 0.6819$. It is generally recommended that the values of b range from 0.5 to 0.8. We chose $b = 0.8$ and varied $k1$ from 0.2 to 1.6 as shown below in the table. At $k1 = 0.4$, A peak in the graph can be observed and the MAP value corresponding to this peak value is 0.6841. We chose the values of b and $k1$ as 0.8 and 0.4 for good MAP value.



3.2.2. Tuning parameters in DFR model:

We tried out various combinations of parameters such as normalization, aftereffect and basic model. We started out the implementation using the default settings given to us which are normalization – H2, Aftereffect – B and Basic model – G. We observe that for normalization – Z, aftereffect – B and basic model – I(F), the results improve very well. So we chose these parameters for the DFR model.

normalization	aftereffect	basic model	map
h3	b	i(f)	0.6683
h2	b	i(f)	0.679
h2	l	i(f)	0.6737
h3	b	g	0.6726
z	b	i(f)	0.6815
z	b	g	0.6755

3.3. Synonyms list

We provided the synonyms list to each core and observed its impact on the model performances. The synonyms list include the same-language synonyms and translations of certain words like U.S.A , Syria ,Russia, Terrorists, Challenges.

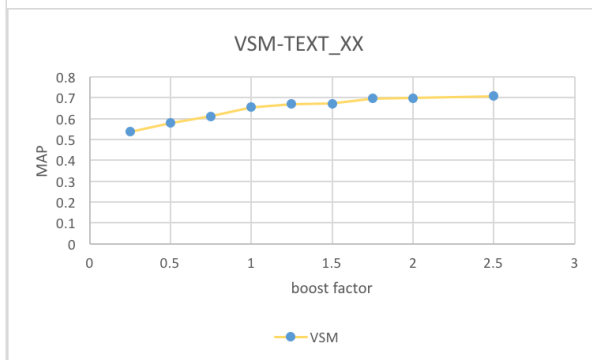
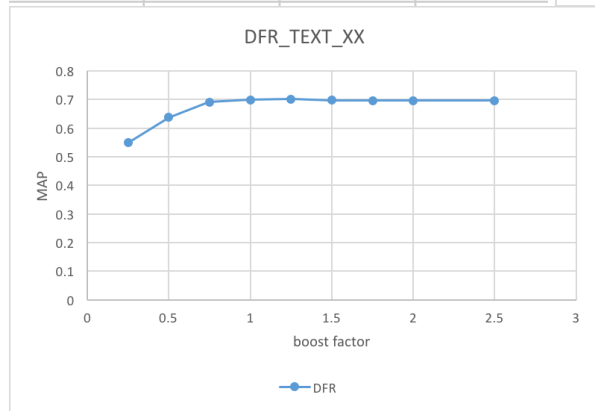
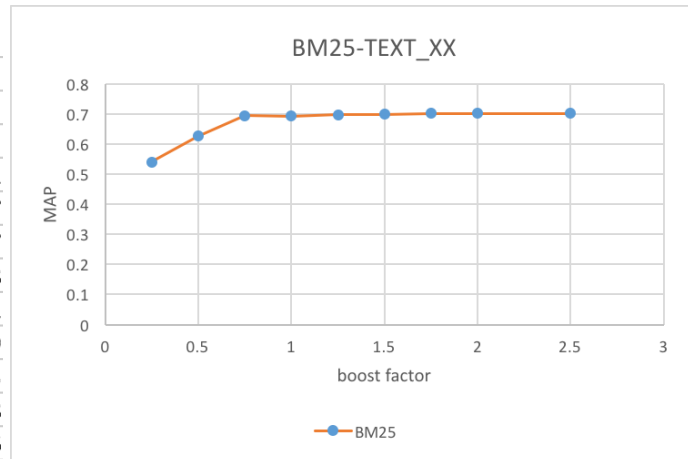
The MAP values obtained after this step for the models DFR , BM25 , VSM are 0.6737, 0.6875, 0.6638.

3.5. Creating a customized query parser based on edismax query parser:

Extended dismax query parser is an improved version of dismax query parser. Since the IR system should have an overall performance improvement, it is better to make changes to the configuration of the cores such that it is not query specific(by performing global changes or boosts) and user is able to fetch results for all possible queries relevant to the document or tweet collection. So it is better to create a custom query parser based on edismax query parser and it is trained over the set of provided queries.

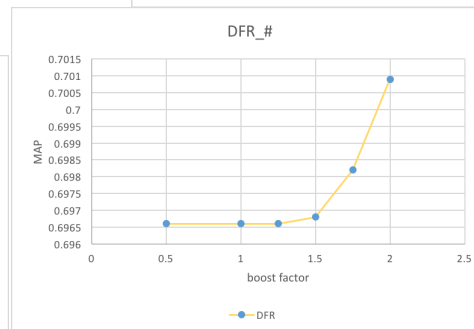
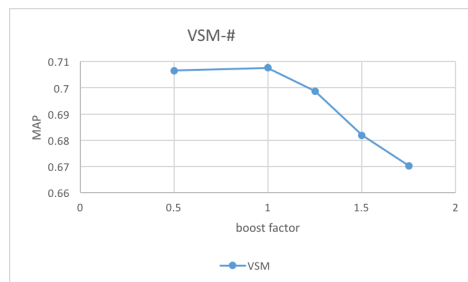
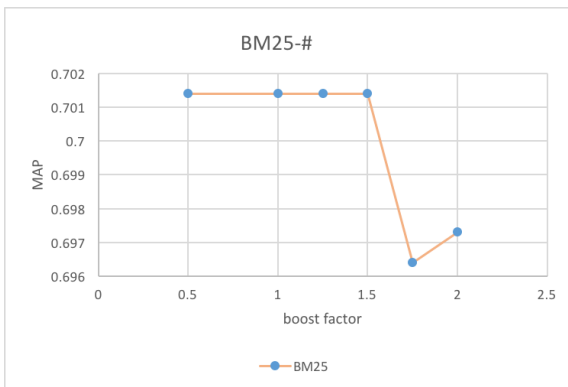
As hashtags are important part of a tweet, it seems reasonable to boost the field tweet_hashtags by specifying a boost factor in the modified query parser. We have used tweets of three different languages – German, English and Russian in this IR system. So we decided to boost the text_en , text_de, text_ru fields according to the original language of the query. We tried out various boost factors to see which boost factor works appropriately for better performance of the models. The effect of **boosting the text_xx fields** (xx = en , de, ru) is illustrated by the following tables and graphs such that the MAP value is improved.

Boost for hashtags = 1			
	MAP		
boost factor	DFR	BM25	VSM
0.25	0.549	0.5406	0.5372
0.5	0.6378	0.6267	0.5787
0.75	0.6913	0.6941	0.6117
1	0.6986	0.6928	0.6548
1.25	0.7009	0.6973	0.6691
1.5	0.6982	0.6984	0.6709
1.75	0.6966	0.7014	0.6963
2	0.6966	0.7014	0.6988
2.5	0.6966	0.7014	0.7076



Effect of boosting tweet_hashtags field after **boosting text_xx fields** to 2.5 is illustrated by the following:

boost for text_xx = 2.5			
	MAP		
Boostfactor	DFR	BM25	VSM
0.5	0.6966	0.7014	0.7066
1	0.6966	0.7014	0.7076
1.25	0.6966	0.7014	0.6988
1.5	0.6968	0.7014	0.6819
1.75	0.6982	0.6964	0.6703
2	0.7009	0.6973	0.6691



It is observed after noticing the above effects on the MAP value that for boost factor for text_xx = 2.5 and boost factor for tweet_hashtags = 1 , the MAP performance improved significantly for VSM and BM25 models(**MAP_VSM = 0.7076, MAP_BM25 = 0.7014**). For boost factor for tweet_hashtags = 2 and boost factor for text_xx = 2.5 , the MAP performance improved significantly for DFR model(**MAP_DFR = 0.7009**).

3.6.CONCLUSION:

After enhancing search engine performance of the models, the following results have been obtained:

BM25:	K1	0.4		
	B	0.8		
DFR:	normalization	Z		
	aftereffect	B		
	basic model	I(F)		
Boost factors:	text_xx	2.5	FOR BM25,VSM	
		2.5	FOR DFR	
	tweet_hashtags	2	FOR DFR	
		1	FOR BM25,VSM	
	MAP_VSM	0.7076		
	MAP_DFR	0.7009		
	MAP_BM25	0.7014		