EVALUATION OF INFORMATION RETRIEVAL MODELS:

Name:Prashi Khurana

UBPersonId : 50316796

1. INTRODUCTION

In this project, we will learn about the different IR models. These models are implemented in Solr. The input data is a dataset of Tweets and the results for the models are compared using the TREC evaluate tool.
First we will try to understand the data flow and the process to get to these models:

1.1 Input Dataset:

The data to be used is Twitter data saved in json format. Three languages are included - English (text_en), German (text_de) and Russian (text_ru).
Example of a tweet in the dataset:


```
{
  "lang": "de",
  "text_de": "RT @JulianRoepcke: ARTIKEL @BILD \n\nRussische Luftschläge in Syrien\nAssad und ISIS auf dem
    Vormarsch\n\nhttp://t.co/PDVxot3CnX http://t.co/a4i…",
  "text_en": "",
  "tweet_urls": [
    "http://www.bild.de/politik/ausland/syrien-krise/assad-isis-syrien-42971016.bild.html"
  ],
  "text_ru": "",
  "id": 653278482517110800,
  "tweet_hashtags": []
}
```


1.2 Indexing the dataset on Solr:

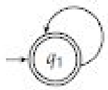
We need to post the dataset on the 'specific' core in Solr to index the dataset.

1.3 Understanding the models theoretically and their hyper parameters:

Language Models:
We need to generate a query from a document model in this case. How do we create this Document Model ?
A language model is a function that puts a probability measure over the strings drawn from some vocabulary. That is, for a language model M over an alphabet Σ.

$$\sum_{s \in \Sigma^*} P(s) = 1$$

After generating each word, we decide whether to stop or to loop around and then produce another word and so the model also requires a probability of stopping in the finishing state.



$P(\text{STOP}|q_1) = 0.2$

the	0.2
a	0.1
frog	0.01
toad	0.01
said	0.03
likes	0.02
that	0.04
...	...

► Figure 12.2 A one-state finite automaton that acts as a unigram language model. We show a partial specification of the state emission probabilities.

But when it comes to scoring documents with this model, we just use the likelihood ratio, which is simply the probability by one model by the probability of the other model.
There are many types of language models, Unigram models, Bigram models, probabilistic context free grammars.

<u>BM25</u>:

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

Here the tf_{td} is the frequency of the term t in a document d, and L_d and L_{ave} are the length of the document d and the average document length for the whole collection. The variable k_1 is a positive tuning parameter that calibrates the document term frequency scaling. A k_1 value of 0 means a binary model(no term frequency), and a large value corresponds to using raw term frequency. B in another tuning parameter which determines the scaling by document length: b =1 corresponds to fully scaling the term weight by the document length, while b=0 corresponds to no length normalization.

<u>DFR</u>:

Also known as Divergence-from-randomness, is a type of probabilistic model. It is basically to test is basically used to test the amount of information carried in the documents. It is based on Harter's 2-Poisson indexing-model. The 2-Poisson model has a hypothesis that the level of the documents is related to a set of documents which contains words occur relatively greater than the rest of the documents. Term weights are being treated as the standard of whether a specific word is in that set or not. Term weights are computed by measuring the divergence between a term distribution produced by a random process and the actual term distribution. DFR models set up by instantiating the three main components of the framework: first selecting a basic randomness model, then applying the first normalization and at last normalizing the term frequencies.
Basic DFR parameters:
basicModel:
G Geometric approximation of the Bose-Einstein
I(F) Inverse Term Frequency Model
Normalization:
H3
H2
Z
AfterEffect:
L Laplace Model

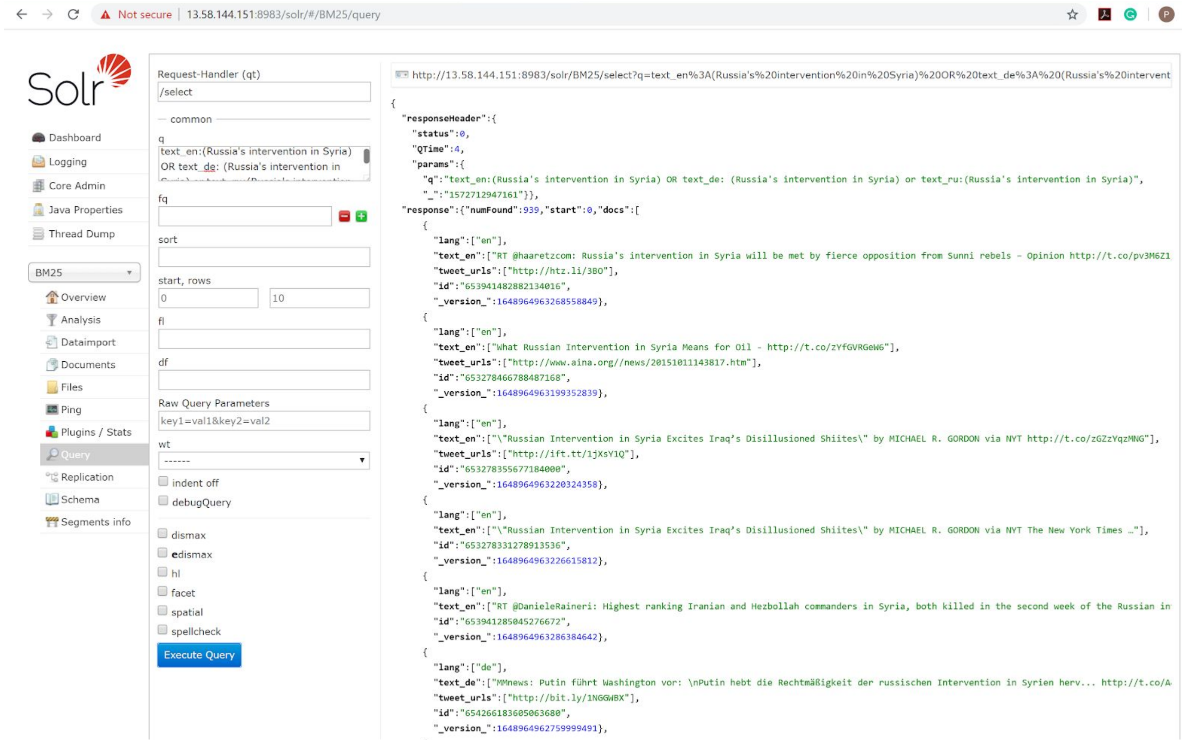
B Ratio of two Bernoulli's process

1.4 Implementing Default IR models:

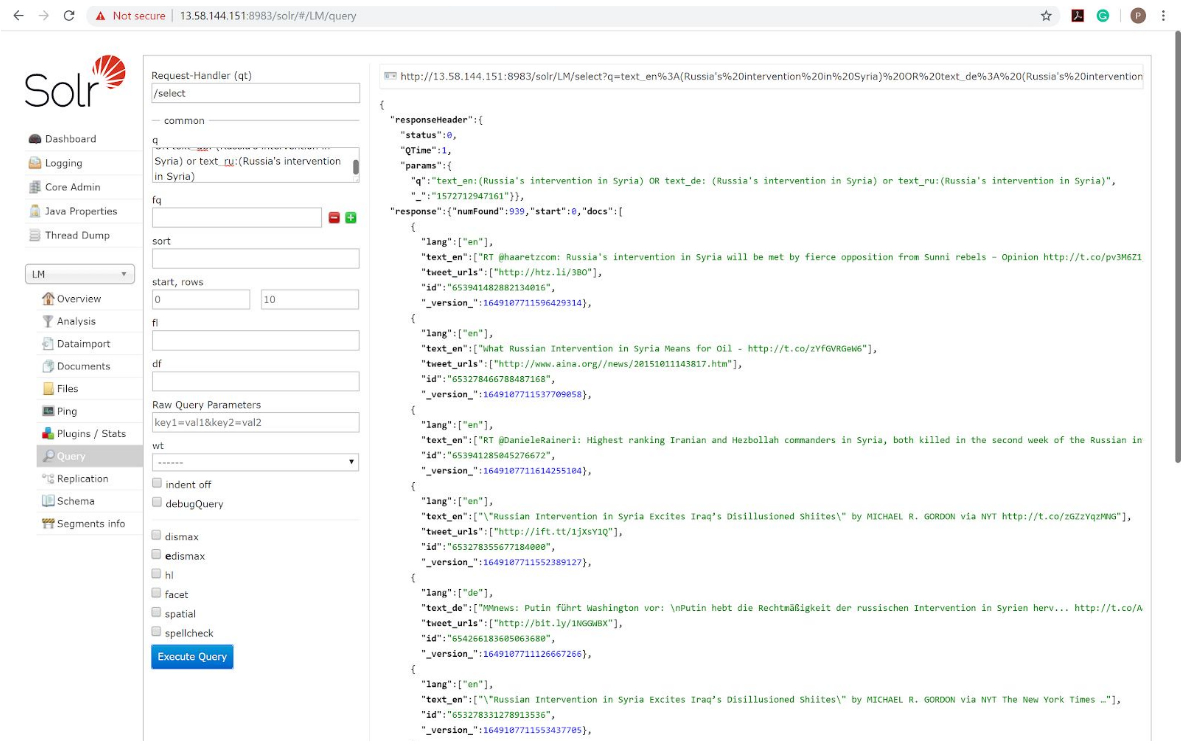
We will be creating a new core for each of the given models. We will be implementing the following model. The default setup for the three models is as follows:
The default setting for each of these models resulted in a very low MAP score:

Sno	Model	Parameters	Param Values	MAP
1	Langauge Model	mu	200	0.6764
2	BM25	k1 b	1.2 0.75	0.6807
3	DFR	normalization afterEffect basicModel	H2 L G	0.6902

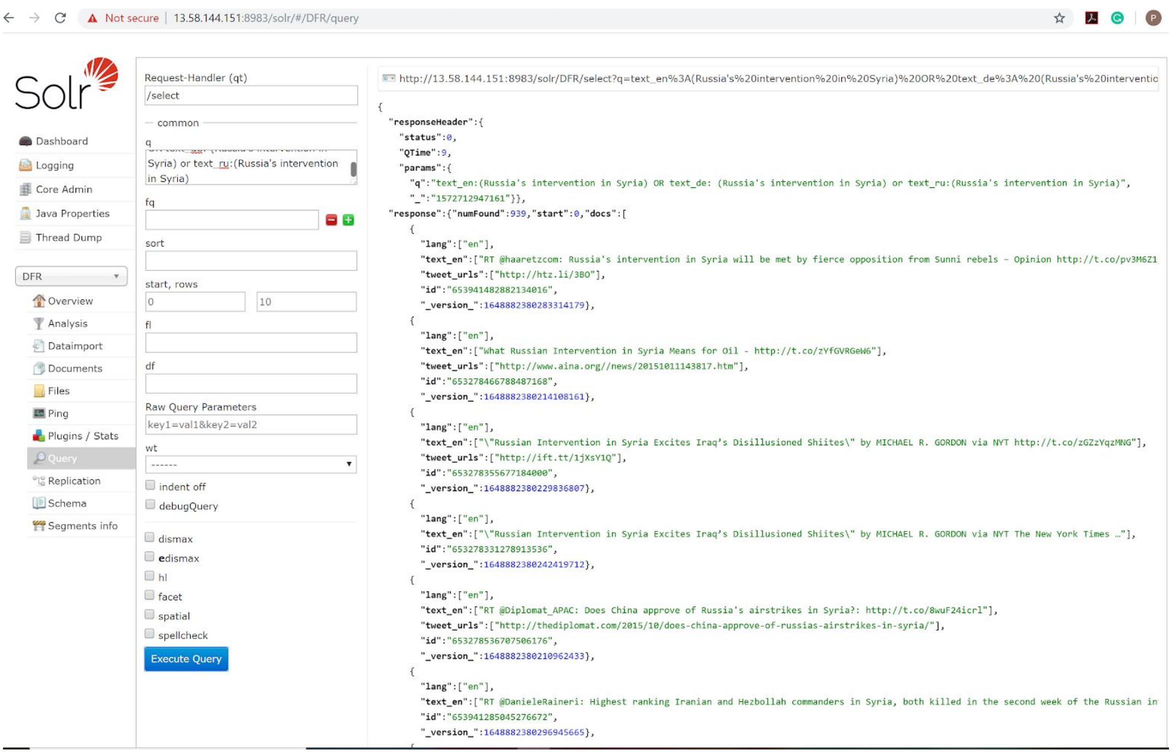
Successful implementation of BM25:



Successful Implementation of LM:



Successful Implementation of DFR:



1.5 Process of getting Results from SOLR

We will be using a python script to get to read the queries one by one and then pass these query to the models and pull the json result and get the relevancy score of the documents based on the query and finally pass this onto the TREC evaluator for each model.
The result of one of the query from a model will be like:
'001 Q0 653941482882134016 1 4.012659 BM25' where 001 is the query Number, Q0 is constant, ignored in TREC, 653941482882134016 is document id, in this case, tweet_id, 1 is the rank of this document for query 001, 4.012659 is the similarity score returned by the IR model in Lucene; BM25 is the model name. There will be 20 such rows.
Similarly for the other models.

1.6 Understanding MAP:

Most standard among the TREC community is Mean Average Precision (MAP), which provides a single-figure measure of quality across recall levels. Among evaluation measures, MAP has been shown to have especially good discrimination and stability. For a single information need, Average Precision is the average of the precision value obtained for the set of top k documents existing after each relevant document is retrieved, and this value is then averaged over information needs. That is, if the set of relevant documents for an information need qj ∈ Q is {d1, . . . dmj} and Rjk is the set of ranked retrieval results from the top result until you get to document dk.

$$\text{MAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_{jk})$$

1.7 Process of getting the TREC results

We will be evaluating our sample_query_output file for each model using TREC eval. The trec evaluation will give us the MAP for all the queries for a particular model and the main aim of this project is to improve the MAP score.

2. IMPROVING MAP SCORE FOR EACH MODEL:

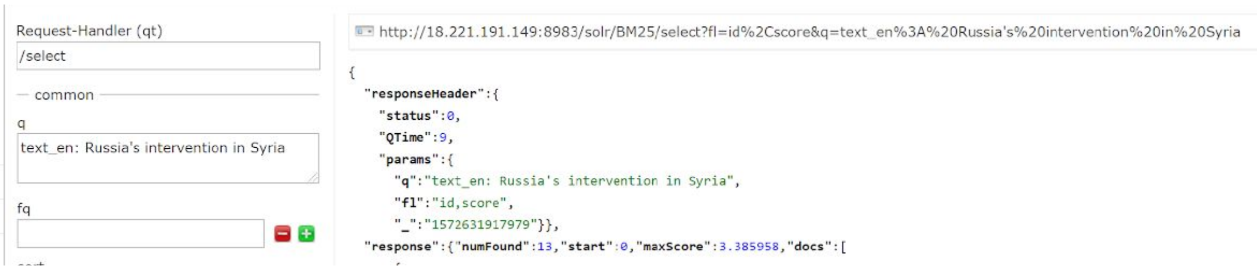
We were provided with a set of test queries which were used to train and tune the hyper parameters.

2.1 Some Basic Query parsing operations before tuning to improve MAP:

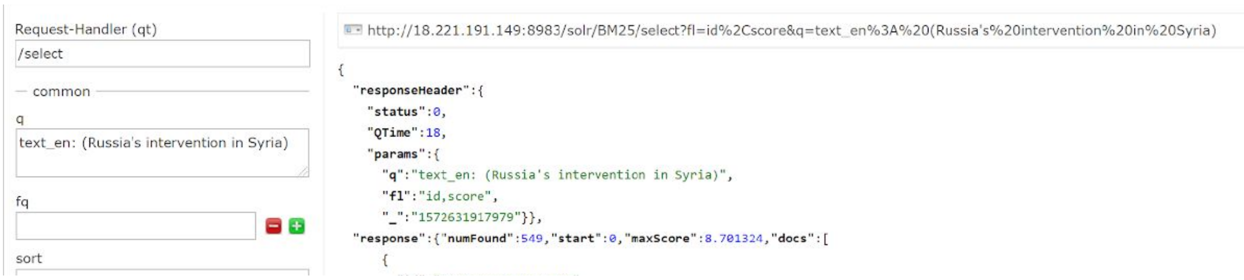
2.1.1 Adding () or \\ across the Query:

As we saw when we debugged the query, Solr was taking the query and comparing only the first word and processing just that. So we added '\\' to each word and then process the query. This ensures all the words are taken into consideration when solr finds documents to be compared.

Before adding () or \\, we get 13 docs.



After adding () or \\: we see we got 549 docs:



2.1.2 Adding all three languages:

We don't know the language of the query being passed. This gives us two options:

1. Either we detect the language of the query

<u>Advantage</u>: We can use this feature to boost the particular language tweet.

<u>Disadvantage</u>: We should not completely ignore the other languages as that might create as issue as people today use mixture of languages in their tweets.

So this is like a trade off, and for this project this is not giving me much of an improvement.

2. Or we use the query with all three options

Ex: text_en= query OR text_de = query OR text_ru = query

Before using All 3 Languages, we get only 549 docs.

After using all 3 languages, we got 939 queries.

2.1.3 Finding Hashtags and Boosting them:

In the particular Query , we might have some hashtags. Hashtags are specified by '#' and we need to look for them in the tweet_hashtags fields as well. So we looked for them and used them to add to the query as a separate parameter.

We saw an improvement of MAP score for DFR. From 0.6902 it went to 0.6927, keeping the default values the same.

2.1.4 Finding retweets and Boosting them:

We can also use the same concept to find retweets or tweets specific to a particular user. Here we get the word used with '@' and look for them and add them to the query but this also does not have a very high impact on this project.

2.1.5 Summary of the methods used:

Improving Models	Before	After	Result
Adding () or ||	13 (no. of docs)	549 (no. of docs)	MAP improved for all three
Language Detect and Boost Adding all 3 lang	549 (no. of docs)	939 (no. of docs)	MAP improved for all three
Hashtags & Retweets Find And Boost Hashtag	0.6902	0.6927	MAP for DFR improved

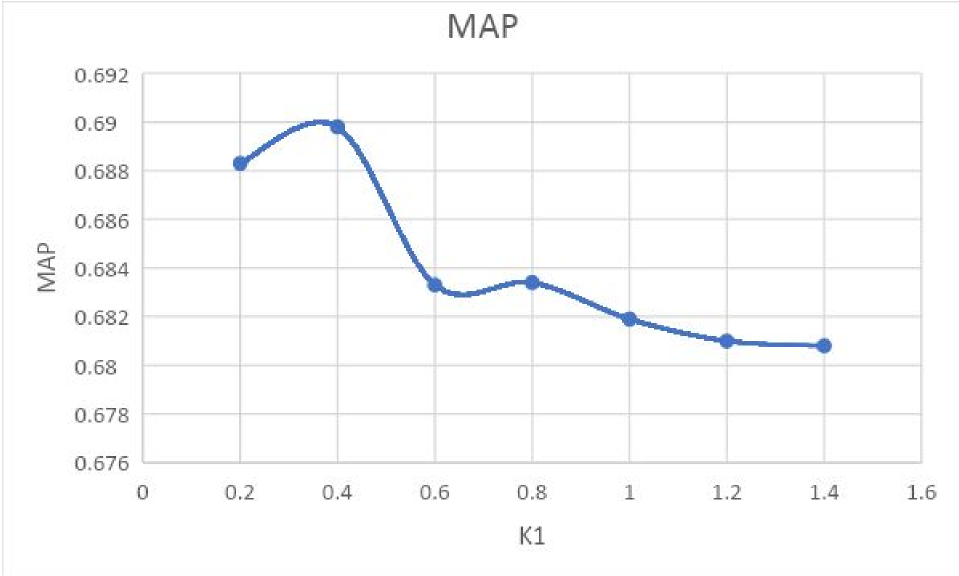
2.2 Results from Tuning parameters for BM25:

Some trials for Bm25**:**

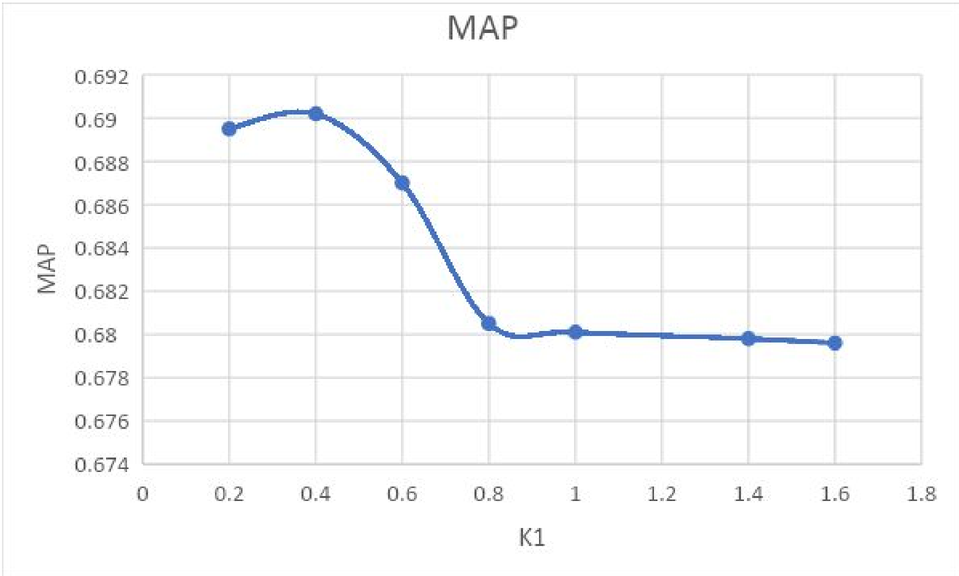
k1	b	MAP
0.2	0.4	0.6886
0.4	0.4	0.6891
0.6	0.4	0.6875
0.8	0.4	0.6871
1	0.4	0.685
1.2	0.4	0.6848
1.4	0.4	0.6844

k1	b	MAP
0.2	0.6	0.6895
0.4	0.6	0.6926
0.6	0.6	0.687
0.8	0.6	0.6805
1	0.6	0.6801
1.4	0.6	0.6798
1.6	0.6	0.6796

For a constant value of b=0.8 and k varying from 0.2 to 1.4 we got the following graph:



For another value of b=0.6 and k=0.4 we got the following graph:



After varying a for a lot of values of b and k1, we get a summary like this:

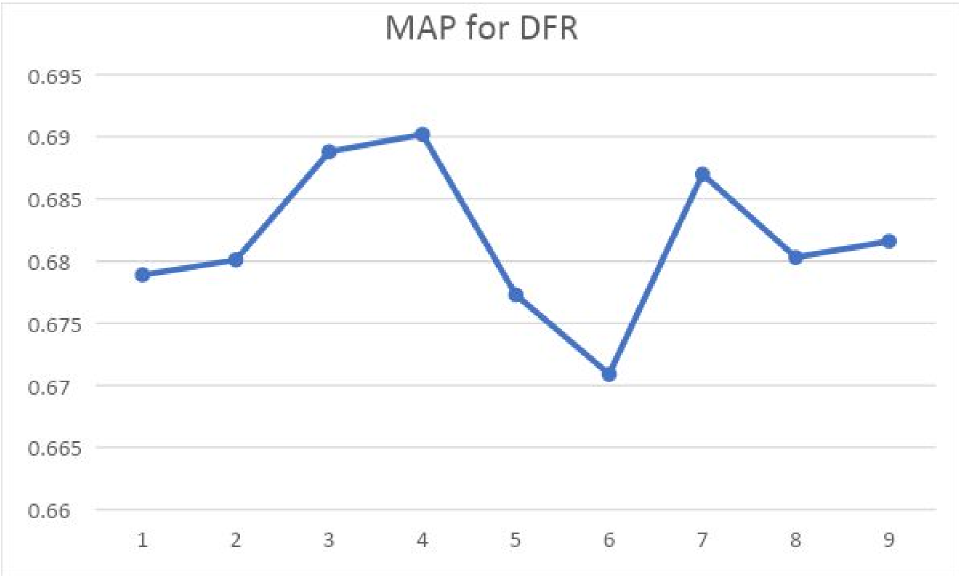
k1	b	MAP
1.4	0.3	0.6873
0.4	0.4	0.6891
0.4	0.6	0.6926
1.2	0.65	0.6803
0.8	0.75	0.683
0.4	0.8	0.6883

As we see the highest score is for the setting of **k1=0.4** and **b=0.6** with a **MAP of 0.6926**.

2.3 Tuning parameters for DFR:

We have 3 parameters to tune here, *normalization*, *afterEffect* and *basicModel*.
The *normalization* takes 3 parameters, Z, H2 and H3
The *aftereffect* takes 2 parameters, B and L
The *BasicModel* also takes 2 parameters, I(F) and G.

Varying all 3 parameters we get the following output:



1	h2	B	i(f)	0.6789
2	h2	B	G	0.6801
3	h2	l	i(f)	0.6888
4	h2	l	G	0.6927
5	h3	B	i(f)	0.6773
6	h3	B	G	0.6709
7	h3	l	i(f)	0.687
8	Z	B	i(f)	0.6803
9	Z	B	G	0.6816

We get the highest **MAP of 0.6902** with **normalization as H2**, **AfterEffect as L** and **BasicModel as G**.

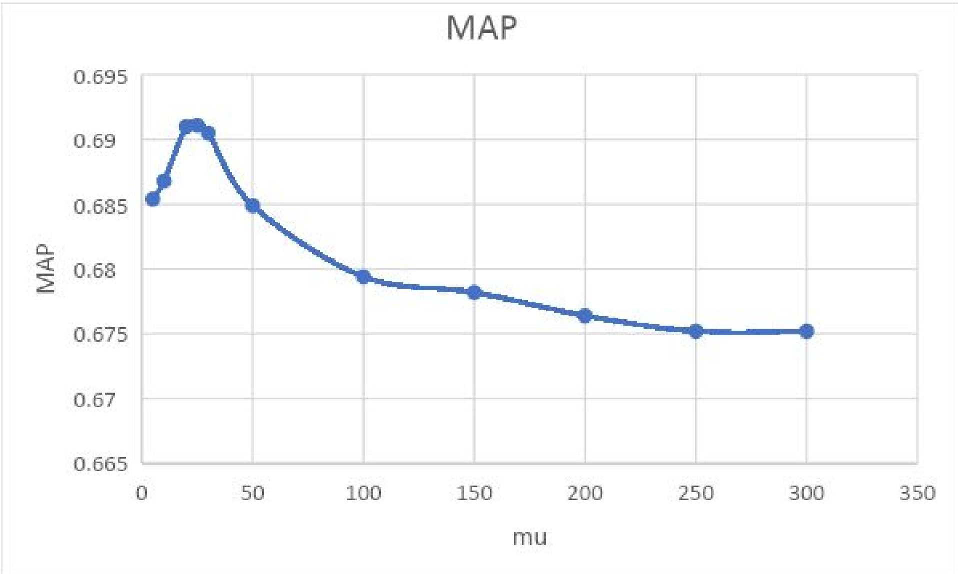
Also as per in section 2.1 we see that after boosting the hashtags, we get a **MAP of 0.6927** for the same setting of the parameters.

2.4 Tuning parameters for LM:

We will be tuning the parameter mu .
Below is the graph where we can see that the peak is at **mu = 25** with a **MAP of 0.6911**

mu	MAP
300	0.6752
250	0.6752

200	0.6764
150	0.6782
100	0.6794
50	0.6849
30	0.6905
25	0.6911
20	0.691
10	0.6868
5	0.6854

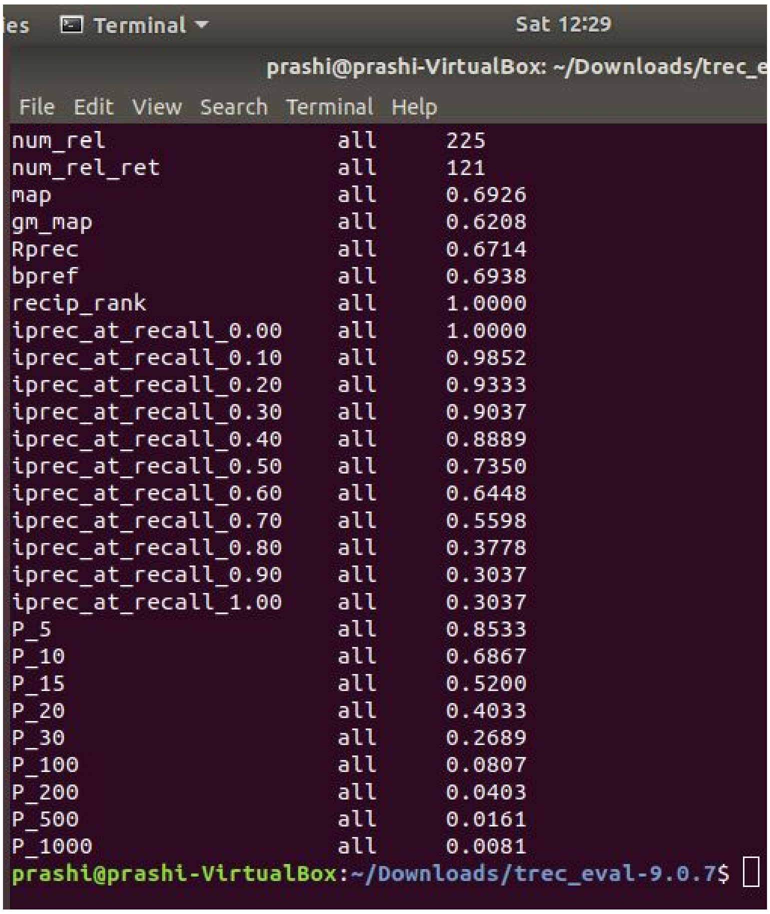


3. RESULTS:

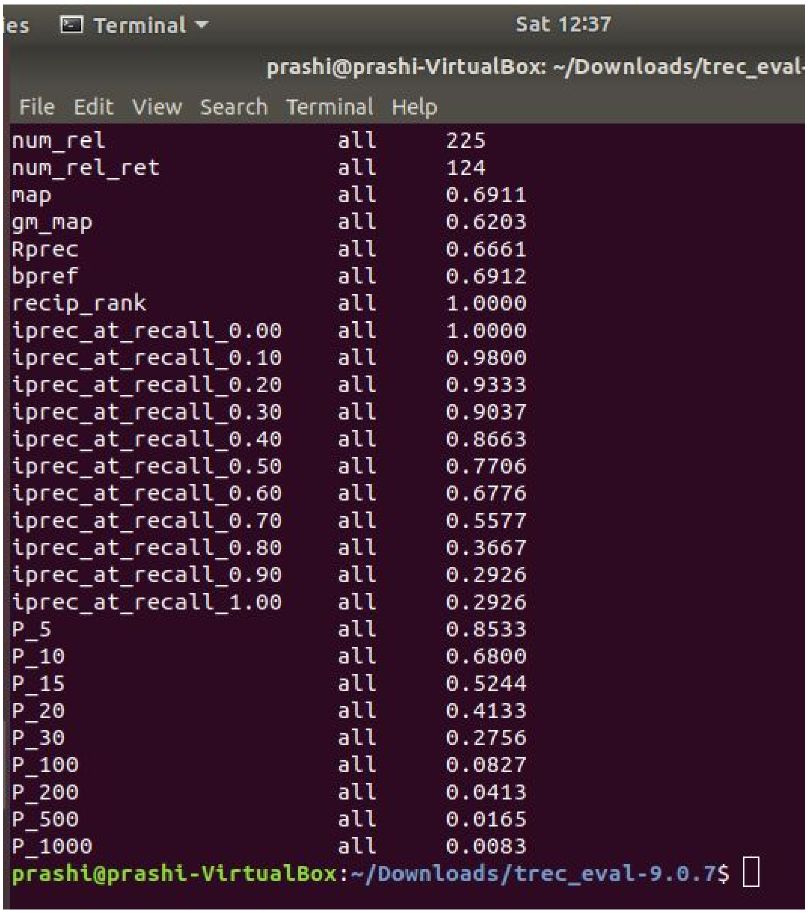
Final Implementation of the best Models:

Core: BM25, with parameters k1:0.4 and b:0.6

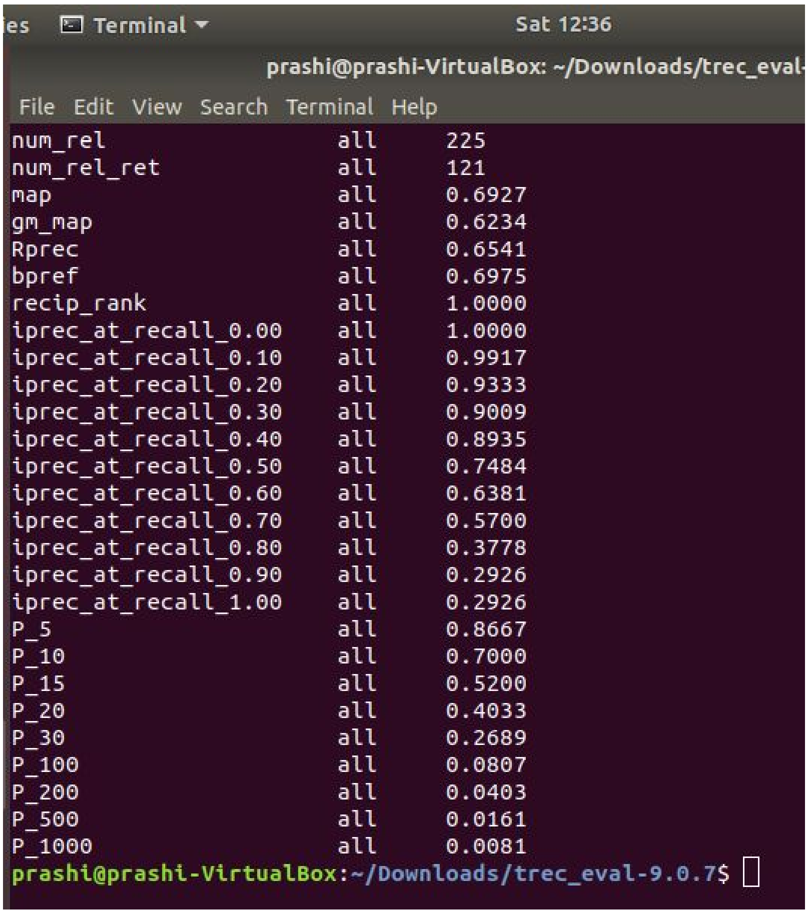
MAP score for BM25:



Core:LM, with mu:25



Core: DFR, with normalization: H2, aftereffect:L, basicModel:G



After all query processing and parsing we can come to the following conclusion:

Sno	Model	Parameters	Param Values	MAP
1	Langauge Model	mu	25	0.6911
2	BM25	k1 b	0.4 0.6	0.6926
3	DFR	normalization afterEffect basicModel	H2 L G	0.6927