Multilingual search system for Tweeter (part B)

Project Report
Tushar Baraiya
UBID:50169977

1. Introduction

A. Problem description

Tweeter data in three language English, German and Russian are given, main aim is to implement different models for information retrieval. Three IR models namely Vector Space model, BM25 model and Language model are required to implement. Furthermore augmentation of the model is required to improve quality of retrieval. Quality of the retrieval is measured using standard trec_eval parameters. Sample queries and relevance judgment is given to tune the model.

B. Strategy

My main strategy is "trial and error" method. Initially, I will change different parameters according to my intuition and check the result. By observing initial results I will change models and parameters of the model. Observation of the result will also help to understand behavior of the retrieval, so observation will play major factor here to improve the retrieval model. I will use standard trec_eval evaluation measure to judge a particular retrieval model.

Improvement of the IR model is the crucial part of the project. I will try several ideas to improve each model. First idea is to play with query parser. Solr provides wide range of query parsers, any of them can be utilized to improve quality. Query expansion can be used to search in different languages. We can also tweak different parameters of similarity models, for example we can change k of the BM25 model. Query boosting can be used to increase the score. Also if needed we can always store extra information during indexing that can help to improve the quality of search. I will try several of these ideas.

C. Test Data

Tweeter data is given in json file format in three different languages. 14 test queries are given for testing of the model, relevance judgment is also given for each query for testing with trec_eval. All the following discussion and result are for the given 14 test queries. In the discussion of different models we will use the average parameters of the all 14 queries.

2. Models and Results

The comparing of the following result is given for same setup for the all three models that mean everything except similarity models is similar. For comparison I have used dismax query parser with boosting for phrase query with 3.0 and three query field text_en, text_de and text_ru. Also tie is given 1.0 value. We will compare results of each model for query 001.

A. Vector space model

Vector space model uses the standard tf-idf waiting for scoring.

Query	Ndcg	map	Bpref	F0.5
001	0.8669	0.4843	0.7175	0.0980

B. BM25

Query	Ndcg	map	Bpref	F0.5
001	0.8380	0.4558	0.7475	0.0980

C. Language model

Query	Ndcg	Map	Bpref	F0.5
001	0.9095	0.6502	0.6850	0.0980

From the above result we can say that each models improves different measurement factors but overall Language model returns good results for most of the parameters!
So, for further improvement I will use language models as basic model and apply all improvement to the language model.

D. Why language model performs better?

I have used LMDirichletSimilarityFactory to implement Language model, this similarity factory implements Language model based on the Jelinek-Mercer smoothing method. Language model uses probabilistic model to calculate probability of query to occur in particular document, this probability distribution is used as model to retrieve data. VSM uses simple tf-idf to calculate score, so it does not consider relation between two different terms. BM25 uses pure probabilistic models to retrieve data. But Language models uses probabilistic model and tf to score a document, it also uses smoothing techniques. S overall performance of the Language model is better.

E. What measure gives a better evaluation of the system and why?

Ndcg gives better evaluation of the system because it gives overall idea of distribution with respect to ideal distribution of the documents.

3. Improving IR system

A. Query parser

From given many options of query parser, I believe **Dismax** parser is suitable for our purpose. This can be seen from given comparison of the result with default parser. For default parser we used OR to pass phrase query.

Parser(for all results)	Ndcg
Default	0.8231
Dismax	0.9143

Another motivation to use dismax parser is because of its different fields that are very useful like pf-phrase query, mm-minimum match, bf-boost function,etc

B. Boosting

Query boosting is very useful tool provided. I am using query boosting with phrase query field (pf) to boost the score of the document which matches all the terms in query. Boosting can be used with text_en to improve score of relevant document because usually text_en have low idf values so we can improve it through boosting. Also we can boost documents with exact term matching. Using this feature significantly improves the ndcg because it will give higher scores to the documents which have all the terms. It also improves MAP because relevant documents are pushed upward on the result.

	Ndcg	Map
Without boosting	0.9011	0.6422
With boosting	0.9143	0.6502

C. Query reformulation

We can expand query in all other language and pass it on different fields.

Expansion(translated query)	Ndcg	Bpref	Map	F0.5
Without	0.9143	0.7129	0.7017	0.1834
With	0.8723	0.7083	0.6632	0.1547

Surprisingly, Query reformulation in different languages is not helping as expected! This drop happens because of the extra terms we have added in the query. These extra terms fetches the unwanted documents from the index, which pushes relevant document back in the ranking. Conclusively this is not a good idea.

D. Customize Query handler

I wanted to pass query of each language on language specific field for example English query on text_en , so I designed one query handler that uses dismax query parser.

```
<requestHandler name="/translate" class="solr.SearchHandler" default="false">
    <lst name="invariants">
      <str name="q">
       (_query_:"{!dismax qf='text_en'  pf='text_all^3' tie=1  v=$eq }"
  OR _query_:"{!dismax qf='text_ru'  pf='text_all^3' tie=1  v=$rqq}"
  OR _query_:"{!dismax qf='text_de'  pf='text_all^3' tie=1 v=$dq}")
      </str>
    </lst>
 </requestHandler>
```

This idea is also a failure because I need to translate query into each language and pass it on every field. Which gives almost similar result as previous case.

E. Changes in Schema.xml

I have made many changes in schema.xml to improve performance of IR system. I made copy field name text_all that contains text of tweet. Only standard tokenizer and lowercase filter factory is used for field type. This field is useful to boost query when phrase matching occurs. I used this field with pf to boost phrase match.
This change has positive effect on results.

	Ndcg	Map
Without boosting of pf	0.9011	0.6422
With boosting of pf	0.9143	0.6502

Another copy field is used to create text_en2, text_de2 and text_ru2 for exact phrase matching. Standard tokenizer and is used for document indexing and for query analyzer I have removed stop words also. This idea is not useful because the document is in 3 different languages so we will have problem with stemming because all three language cannot be stemmed with same algorithm! Also we will have problem with removing stop words because it is quite possible that one stop word is not a stop word for another language!

3. Conclusion

Conclusively Among the 3 models, Language Model provides best overall result. And for improvement of model boosting and using proper parser (dismax) helps a lot. Query reformulation (translation) does not help. So main features of my optimal model is LM model,dismax query parser, query boosting , copy fields to boost exact match as pf.