CSE 435/535 Information Retrieval
Fall 2015
Project Part B: Evaluation of IR models

Due Date: 23:59, Nov 13th, 2015

Overview

The goal of this project is to understand and implement various IR models, use different evaluation measures to evaluate the IR system and improve the search result based on the understanding of evaluation. In particular, you are given twitter data in three languages, English, German and Russian on topic European refugee crisis, 14 sample queries and the corresponding relevance judgement. You will index the given twitter data and 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, you are asked to improve the performance of the system on measures such as F0.5, MAP, nDCG and etc.

The following sections describe the various tasks involved, evaluation criteria and submission guideline.

Twitter Data

[provided files: train_raw.zip, train.json]

The data given is Twitter data collected in json format on topic European refugee crisis in English, German and Russian. You will be given two different files **train_raw.zip** and **train.json**.

train_raw.zip: This file includes 38 json files with 100 tweets per file. Each tweets is in its raw json format.

train.json: This file contains the **SAME** tweets as train_raw.zip, only with some fields extracted. Sample tweet format is as follows:

```
{
  "lang":  ,
   "id": ,
  "text_de": ,
   "text_en": ,
   "text_ru": ,
   "created_at": ,
   "tweet_urls": [ ],
   "tweet_hashtags": []
}
```

You are free to use either one of these data. If you want to create your own tweet format for better search result, use the raw tweets to extract the fields that is useful to you. **NOTE**: if you are using raw json files, there are about 32 tweets are duplicated among the files, i.e., with the same id. You will need to delete the duplicates in order to use TREC_eval for further evaluation. If you are using train.json, the tweets are already unique, so you don't need to check duplication.

Index

In this step, you will need to index the data as you have done in part A. Note that, whenever you change the Solr schema.xml file or solr config files, you will need to reindex.

Implementing IR models

[provided files: queries.txt, qrel.txt, sample_query_output.txt]

In this step, you will need to implement Vector Space Model (VSM) and BM25 in Solr. Please read the link

http://wiki.apache.org/solr/SchemaXml#Similarity

to understand how you can choose different similarity function in Solr.
http://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html?is-external=true
gives you the idea on what is Solr's default similarity. Check
https://wiki.apache.org/solr/SolrPlugins#Similarity for more information on using custom similarity function. http://stackoverflow.com/questions/20428709/solr-custom-similarity is the discussion on how to use this in solr.

Additionally,
http://lucene.apache.org/solr/4_0_0/solr-core/org/apache/solr/search/similarities/package-summary.html
gives you all the similarity functions you can choose from Solr.

You might need to use different methods to improve the retrieval system result by examining the TREC_eval result. Details explained in the section ***Improving IR System***.

Input Queries for search system

You are provided with 14 sample queries (**queries.txt**) and corresponding manually judged relevance score (**qrel.txt**).

queries.txt, includes 14 sample queries. One query per line. Each line has format

query_number query_text

For example,

001 Russia's intervention in Syria

Your retrieval result is mainly based on the query_text.

qrel.txt, includes manually judged relevance score. Format as shown below

query_number 0 document_id relevance

For example,
001 0 653278482517110785 0

Search result of the Solr system
You are asked to use json format as Solr search result, which include at least tag: **id** and **score.**
For example, you can use
http://ruhan.koding.io:8983/solr/partb/select?q=*%3A*&fl=id%2Cscore&wt=json&indent=true&rows=1000
to get the score and id(change the username ruhan to your own username).
For more query parameter, please check
https://cwiki.apache.org/confluence/display/solr/Common+Query+Parameters

After this, you will need to post-process your output into a standard TREC_eval format as described below to evaluate your system performance.

Output
The final output of the search system should be a ranked list of document as returned by the retrieval system. It should have the following format,

query-number Q0 tweet_id rank similarity_score model_name

For example,

001 Q0 653278466788487168 0 0.22385858 default

where,
001 is the query number;
Q0 is a constant, ignored in TREC evaluation;
653278466788487168 is the document id. In this case, tweet_id;
0 is the rank of this document for query 001;
0.22385858 is the similarity score returned by IR model VSM, which is default in Lucene;
default is the model name you used.

A sample output file is provided in file **sample_query_output.txt**.

TREC Evaluation

[provided files: default.txt]

In this part, you will be using TREC_eval program. You can download the latest version from http://trec.nist.gov/trec_eval/. After downloading, read the **README** file carefully. One of the basic command is

trec_eval -q -c -M1000 official_qrels submitted_results

For example you can use following command to evaluate the sample query output file.

trec_eval -q -c -M 1000 qrel.txt sample_query_output.txt

This command will give you a number of common evaluation measure results. You can also use **-m** option to specify the measure you prefer. For example

trec_eval -q -c -M 1000 -m ndcg qrel.txt sample_query_output.txt

This command will give you nDCG measure result for each query followed by overall performance.

For more information on how to use or interpret the result, go to
http://www-nlpir.nist.gov/projects/t01v/trecvid.tools/trec_eval_video/A.README

A sample TREC_eval output file is provided in file **default.txt**, which is the result of running the above command.

Improving IR system

This is the main part of this project. Together with your query result, query texts, ground truth result and the TREC_eval result, you might gain an intuition on the performance of your IR system. We will be evaluating your system based on a few measures, including F0.5, nDCG, MAP and etc. Details explained in the following section. Here is a list of things you might want to consider to improve your evaluation score.

1. Examine the query provided and the given ground truth result. Why some tweets are ranked higher than the others? Is it different for different language?
2. Do you need to do advanced **query processing** to improve the result? For example, boosting the query based on different fields? Expand the query? Use different query parser? More details can be found in https://cwiki.apache.org/confluence/display/solr/The+Standard+Query+Parser
3. If the provided query parser doesn't address your problem, you can implement your own query parser and use it in Solr. Read https://wiki.apache.org/solr/SolrPlugins for more details on how to do this.

4. Do you need to have better index? For example, do you need to have additional fields to use additional analyzer and tokenizer to achieve better query result? For example http://wiki.apache.org/solr/SolrRelevancyFAQ#How_can_I_make_exact-case_matches_score_higher
5. Do you need to tweak the parameters of the IR model to make it more suitable to the query? For example, in BM25 model, there are two parameters you can set up. What is the meaning of these parameters and how to tweak it?
6. Do you need to use any filters for query processing?
7. Understand the measure itself. For example, to improve F0.5 measure result, do you need to pay more attention on improving precision or recall?

Here is some documents that might help you with improving your system.

1. https://lucene.apache.org/core/2_9_4/api/all/org/apache/lucene/search/Similarity.html
2. http://wiki.apache.org/solr/SolrRelevancyFAQ#How_can_I_make_exact-case_matches_score_higher
3. https://lucene.apache.org/core/5_3_0/core/org/apache/lucene/search/similarities/BM25Similarity.html

Grading criteria

About one week before the deadline, you will be given a new set of queries. You will be asked to provide the search result and this will be evaluated using measures F0.5, MAP, nDCG. We will be choosing **one more** measure as a wildcard measure. This will be announced later.

Grading will be based on the measure results of your test dataset on VSM and BM25 and report.
You will have to create one collection for each IR model.

Your report should at least include the following contents:

1. Analyze TREC_eval result on different models. Which one performs better overall and why?
2. What have you done to improve the performance?
3. How does the performance change when using different setups? You can use a graph to illustrate how the system performance changed.
4. What measure gives a better evaluation of the system and why?

What to submit?

NOTE: It is your responsibility to follow the submission guideline. Since we will be using automatic grading, the name of the files should be followed strictly.

1. Your report in **pdf** format. File name is **report.pdf** (no other file format is allowed)

2. Your query address for each test query and each IR model. Details will be given later.
3. Your source file (java files) of any customized functions in src folder.

Compress these files into a tar file. File name is project_partb_[ubitname].tar (no other compressed format is allowed)

For example my ubit name is ruhan then I should use following command to submit.

submit_cse535 project_partb_ruhan.tar

Choose cse435 or cse535 based on your own course level.