CSE 435/535 Information Retrieval (Fall 2016)

Project 3: Evaluation of IR models

Due Date: November 10th (Thursday) 2016, 23:59 pm

Overview

The goal of this project is to implement various IR models, evaluate the IR system and improve the search result based on your understanding of the models, the implementation and the evaluation.

In particular, you are given twitter data in three languages, English, German and Russian on topic European refugee crisis, 20 sample queries and the corresponding relevance judgement. You will index the given twitter data and implement Vector Space Model, BM25 and Divergence from Randomness Model based on Solr and evaluate the three sets of results using TREC_eval program. Based on the evaluation result, you are asked to improve the performance in terms of the measure Mean Average Precision (MAP). In this project, as you work and play with Solr, you may need to refer to Solr Reference Guide (http://archive.apache.org/dist/lucene/solr/ref-guide/apache-solr-ref-guide-6.2.pdf) frequently to complete your tasks.

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

Section 1: 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.

Section 2: Implementing IR models

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

Index

In this step, you will need to index the data as you have done in project 1. Note that, whenever you change the Solr schema.xml file or solr config files, you will need to reindex.

Various IR models

In this step, you will need to implement Vector Space Model (VSM), BM25 and Divergence from Randomness Model (DFR) in Solr. Please check the following links:
https://wiki.apache.org/solr/SolrPlugins#Similarity, http://wiki.apache.org/solr/SchemaXml#Similarity on how you can specify and customize different similarity functions in Solr;
http://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html?is-external=true on Solr's default similarity;
http://stackoverflow.com/questions/20428709/solr-custom-similarity on details of how to incorporate your own similarity 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, which means that very likely you don't need to implement an IR model from scratch.

Input Queries

You are provided with 20 sample queries (**queries.txt**) and corresponding manually judged relevance score (**qrel.txt**).

queries.txt, includes 20 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 is as shown below

query_number 0 document_id relevance

For example,

001 0 653278482517110785 0

Query result of Solr

The query result of Solr can be specified into json format, which include at least tag: **id** and **score.** For example, you can use http://localhost:8983/solr/corename/select?q=*%3A*&fl=id%2Cscore&wt=json&indent=true&rows=1000 to get the score and id (change the "localhost" as your hostname and "corename" as the name of your Solr core). For more query parameter, please check https://cwiki.apache.org/confluence/display/solr/Common+Query+Parameters

The query result should be processed into below format to accommodate the input format of TREC evaluation program. A Python script (**json_to_trec.py**) is provided to help you accomplish this task.

The final result 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**.

Section 3: TREC Evaluation

[provided files: qrel.txt, 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**.

Section 4: Improving IR system

This is the main part of this project. Together with your training queries, query results, ground truth judgements and the TREC_eval result, you might gain an intuition on the performance of your IR system. We choose the measure MAP as main objective to improve. 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. Understand the measure itself. How to improve MAP?
3. 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? Use any filters for query processing? More details can be found in https://cwiki.apache.org/confluence/display/solr/The+Standard+Query+Parser
4. 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.
5. 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
6. 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?
7. Do you need to customize your own similarity functions or make some adaption?

You are not bounded with these considerations and definitely you can make other meaningful attempts and report them to us.

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

Section 5: Grading Criteria and Submission

The total points of this project is 15. We will evaluate your work within three aspects:
first, if you have successfully implemented those three models with default settings, you get 5 points (1+2+2). The default setting for each model can be found at http://lucene.apache.org/solr/4_0_0/solr-core/org/apache/solr/search/similarities/package-summary.html
For the DFR model, please choose "BasicModelG" plus "Bernoulli" first normalization plus "H2" second normalization.
Second, there is 5 points given based on the work you have done to improve performance. We will read your report and judge its quality.
Third, the remaining 5 points is given based on the performance (mainly MAP) of your best-effort systems (three models) on test queries. We will quantify the performance of the whole class and the top 20% will get full 5 points, 4 points for next quantile, etc.

About one week before the deadline, you will be given 10 test queries. You will be asked to provide the query results in the same format of **sample_query_output.txt** for each query, each model.

You also need to submit a report, which include the following contents:

1. Describe how to implement each model in Solr and provide screenshots on your key implementation and results to demonstrate that you have successfully implemented them.
2. What have you done to improve the performance in terms of MAP (and maybe also other measures)? Please list what you have done one by one and present why you do this, what the effect is before and after your intervention. You are suggested to use tables or plots to make the comparison informative and clear.

Your report is suggested to be brief and technical-oriented, because we don't judge it based on its length.

How 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. A pdf named "report.pdf". This is your report in **pdf** format. In your report, please include the name and UBIT name of each member.
2. A folder named "VSM", in which are 10 .txt files. Those .txt files are named 1, 2, … 10, respectively, corresponding to the test query 1,2,...10.
3. A folder named "BM25", in which are 10 .txt files. Those .txt files are named 1, 2, … 10, respectively, corresponding to the test query 1,2,...10.
4. A folder named "DFR", in which are 10 .txt files. Those .txt files are named 1, 2, … 10, respectively, corresponding to the test query 1,2,...10.
5. A folder named "src", in which are your source files (include raw java code, your schema for each model, and any other customized sources). This folder can be empty if you don't have those.

Compress these files into a tar file. File name is IR_P3_Team[teamnumber].tar (no other compressed format is allowed). Examples are IR_P3_Team1.tar, IR_P3_Team58.tar. Submit the file on Timberlake server using **submit_cse535 IR_P3_Team1.tar** command**.** Choose cse435 or cse535 based on your own course level.