CSE 435/535 Information Retrieval (Fall 2018)

Project 3: Evaluation of IR models

Due Date: November 15th (Thursday) 2018, 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.

You are given twitter data in three languages - English, German and Russian, 15 sample queries and the corresponding relevance judgement. You will index the given twitter data by Solr 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).

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

Section 1: Dataset

[provided file: train.json]

The data given is Twitter data saved in json format, **train.json**. Three languages are included - English (text_en), German (text_de) and Russian (text_ru).

train.json: This file contains the tweets with some fields extracted from raw data. Sample tweet format is as follows:

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


```
{
  "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": []
}
```

Section 2: Implementing IR models

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

Index

In this step, you will need to index the data as you have done in project 1. Refer to resources on Piazza for more instructions and guidance from project 1.

Various IR models

In this step, you will need to implement Vector Space Model (VSM), BM25 and Divergence from Randomness Model (DFR) in Solr.

Here are some useful links for your reference:

- For luceneMatchVersion earlier than 6.0, VSM is used as the default similarity function, refer to the link for more information http://lucene.apache.org/core/7_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html?is-external=true
- For luceneMatchVersion 6.0 and later, BM25 is used as the default.
- Specify and customize different similarity functions in Solr:
    - http://lucene.apache.org/solr/guide/7_5/other-schema-elements.html#OtherSchemaElements-Similarity
    - http://wiki.apache.org/solr/SchemaXml#Similarity
- More similarity functions you can choose from for Solr, which means that very likely you do NOT need to implement an IR model from scratch: http://lucene.apache.org/solr/7_0_0/solr-core/org/apache/solr/search/similarities/package-summary.html

Input Queries

You are provided with 15 sample queries (**queries.txt**) and corresponding manually judged relevance score (**qrel.txt**).

queries.txt, includes 15 sample queries. One query per line. Each line has the 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 results from 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 "localhost" as your hostname and "corename" as the name of your Solr core). For more query parameters, please check
https://cwiki.apache.org/confluence/display/solr/Common+Query+Parameters

The query results 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 documents 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**.

NOTE: For final submission, we ask you to restrict the (maximum) number of returned documents as **20**, i.e., in each query url, add "row=20".

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 commands 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

Together with your training queries, query results, ground truth judgements and the TREC_eval result, by now 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 could try to improve your evaluation score.

1. Understand the measure itself. How to improve MAP?
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, say translate the query into other languages? 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

3. 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
4. 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?

In your report, describe in detail what are the methods you use to improve the model and your observations.

Section 5: Grading Criteria and Submission

The total points of this project are 15. We will evaluate your work within three aspects:

1. If you have successfully implemented those three models with default settings, you get 5 points (**3+3+3**). The default setting for each model can be found as http://lucene.apache.org/solr/7_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.
2. There are **3** points given based on the work you have done to improve performance. We will read your report and judge its quality.
3. The remaining **3** points are given based on the overall performance (mainly MAP) of your best-effort systems (among three models) on test queries. We will quantify the performance of the whole class and the top 30% will get full 3 points, 2 points for next quantile, etc.

About one week before the deadline, you will be given 5 test queries. You will be asked to provide the top 20 query results in the same format as **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
2. implementation and results to demonstrate that you have successfully implemented them.

3. 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
4. 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 your name and UBIT name.
2. A folder named "VSM", in which are 5 .txt files. Those .txt files are named 1, 2, ... 5, respectively, corresponding to the test query 1,2,...5. Each .txt file contains the top 20 documents returned by your model.
3. A folder named "BM25", in which are 5 .txt files. Those .txt files are named 1, 2, ... 5, respectively, corresponding to the test query 1,2,...5. Each .txt file contains the top 20 documents returned by your model.
4. A folder named "DFR", in which are 10 .txt files. Those .txt files are named 1, 2, ... 5, respectively, corresponding to the test query 1,2,...5. Each .txt file contains the top 20 documents returned by your model.
5. A folder named "src", in which are your source files (include your schema for each model, and any other customized sources).
   NOTE: naming convention for the schema files: schema-vsm.xml, schema-bm25.xml, schema-dfr.xml

Compress these files into a zip file. File name should be **UBITName_project3.zip** (no other compressed format is allowed). Submit the file on Timberlake server. Choose cse435 or cse535 based on your own course level.

NOTE: Late submissions will NOT be accepted. The deadline is firm (i.e., November 15th 2018, 23:59 PM (EST)), if your timestamp is 12:00 AM, it is a late submission. Please start early.

FAQs and Tips:

1. In this project, as you work and play with Solr, you may need to refer to Solr Reference Guide https://lucene.apache.org/solr/guide/7_4/index.html) frequently to complete your tasks.

2. For macOS, if you encounter the following error when installing trec_eval:
   ```
   invalid active developer path
   (/Library/Developer/CommandLineTools), missing xcrun at:
   /Library/Developer/CommandLineTools/usr/bin/xcrun
   ```
   Refer to https://apple.stackexchange.com/questions/254380/macos-mojave-invalid-active-developer-path for more details.

3. For macOS, if you encounter the following error when running trec_eval:
   ```
   trec_eval: command not found
   ```
   Refer to https://www.reddit.com/r/informationretrieval/comments/58luyt/need_help_running_the_trec_eval_program/ for a solution.

4. **Should I work with schema or schema-less mode?**
   - You can either work with or without schema, the performance won't be different.

5. **Which Solr version should I use?**
   - You can use any version as you prefer. For example, either solr-6.6.5 as in project 1, or Solr-7.4.0 as in project 2.

6. **Do I need to implement the models on AWS or local machine?**
   - It's up to you, you can either use your AWS account or you can also conduct the experiments locally.