CSE 435/535 Information Retrieval (Fall 2021) Project 3: Evaluation of IR models
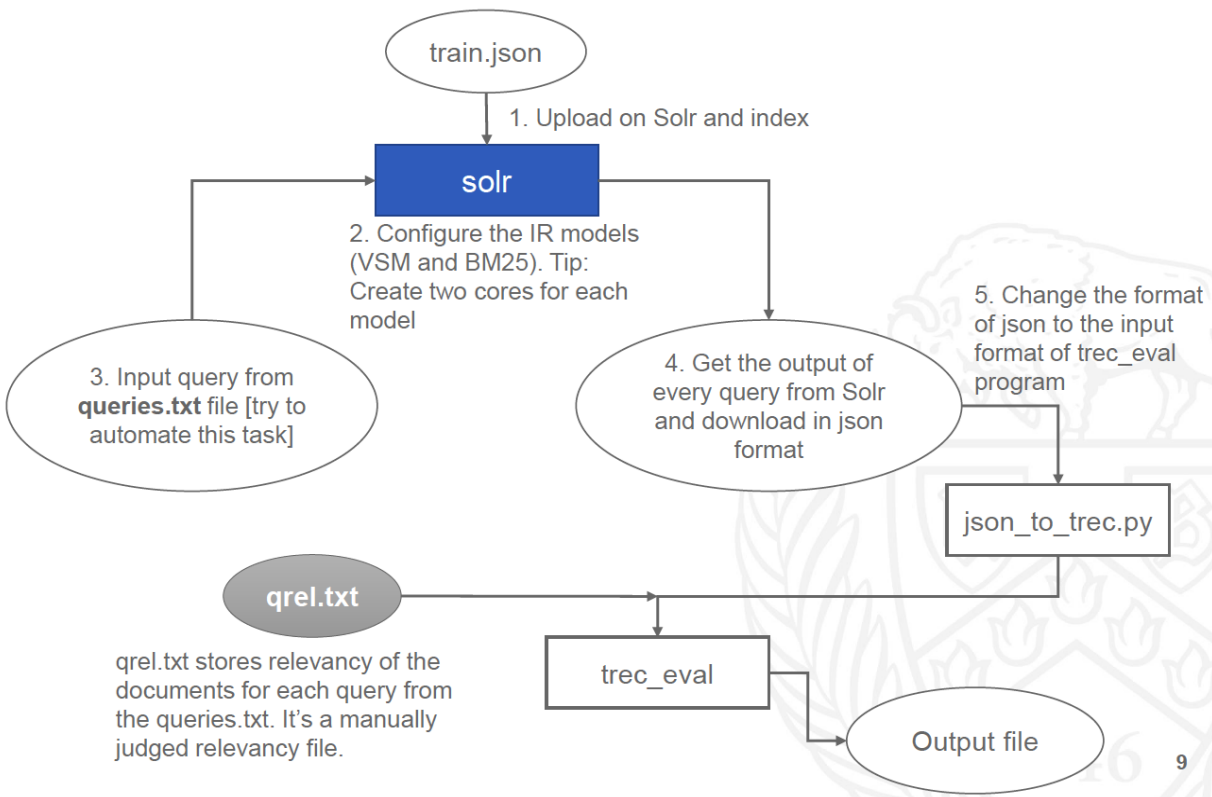
Due Date: November 5th (Friday) 2021, 11:59 pm

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 judgements. You will index the given twitter data using Solr, 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 programatically improve the performance in terms of the measure Mean Average Precision (MAP).

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

train.json

1. Upload on Solr and index

solr

2. Configure the IR models (VSM and BM25). Tip: Create two cores for each model

3. Input query from **queries.txt** file [try to automate this task]

4. Get the output of every query from Solr and download in json format

5. Change the format of json to the input format of trec_eval program

json_to_trec.py

qrel.txt

qrel.txt stores relevancy of the documents for each query from the queries.txt. It's a manually judged relevancy file.

trec_eval

Output file

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": 6532784482517110800,
  "tweet_hashtags": ☐
}
```

Section 2: Implementing IR models

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

Index

In this step, you will need to index the data as you have done in project 1.

Various IR models

In this step, you will need to implement Vector Space Model (VSM) and BM25 (Note that Solr version 6.0 and above by default uses BM25 model). In Solr, these models are implemented through a predefined class called "Similarity".

Here are some useful links for your reference:

- All similarity classes that you can choose from Solr, which means that very likely you do NOT need to implement an IR model from scratch:

https://lucene.apache.org/core/7_7_3/core/org/apache/lucene/search/similarities/package-summary.html

- To specify and customize different similarity functions in Solr Schema:

  - https://solr.apache.org/guide/7_5/other-schema-elements.html#similarity

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 following format:

<div align="center">

query_number query_text

</div>

  For example,

<div align="center">

001 Russia's intervention in Syria

</div>

 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,

<div align="center">

001 0 653278482517110785 0

</div>

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 a query link like this:
http://localhost:8983/solr/corename/select?q=*%3A*&fl=id%2Cscore&wt=json&indent=true&rows=20
to get the score and id (Note: change the "localhost" as your hostname and "corename" as the name of your Solr core). For more query parameters, please check
https://solr.apache.org/guide/7_5/common-query-parameters.html

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 documents as returned by the retrieval system. It should have the following format,

<div align="center">

query-number Q0 tweet_id rank similarity_score model_name

</div>

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 BM25, which is default in Lucene; **default** is the model name you used.

A sample file is provided in file **sample_trec_input.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 "rows=20".

Section 3: TREC Evaluation

[provided files: qrel.txt, sample_trec_output]

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_trec_input.txt

This command will give you a number of common evaluation measure results.

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.READM

E A sample TREC_eval output file is provided in file **sample_trec_output.txt**.

Section 4: Improving the 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://solr.apache.org/guide/7_5/the-standard-query-parser.html
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?

Section 5: Grading Criteria and Submission

The total points of this project are 10. We will evaluate your work within 2 aspects:
1. If you have successfully submitted and implemented two models with default settings, you get 6 points (3+3). The default setting for each model can be found at https://lucene.apache.org/core/7_7_3/core/org/apache/lucene/search/similarities/package-summary.html
2. The remaining 4 points are given based on the performance (mainly MAP) of your best effort systems (among 2 models) on test queries. We will quantify the performance of the whole class and the top 30% will get full 4 points, 3 points for the next quantile, etc.

About one week before the deadline, you will be given **5 test queries**. You will be asked to provide the query results in the same format of **sample_trec_input.txt** for each query, each model.

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 folder named "VSM", in which there 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.
2. A folder named "BM25", in which there 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 "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

Compress these files into a zip file. File name should be **UBITName_project3.zip** (no other compressed format is allowed). For example: **sougatas_project3.zip**. Submit the file on Timberlake server. Choose submit_cse435 or submit_cse535 based on your course level.

Dry Run
You will have an opportunity to validate your submission through a dry run on **3rd Nov, 8 PM ET**. Note that this dry run will only check if the naming convention is correct and we are able to run TREC on your submission successfully. <u>MAP scores will NOT be evaluated during dry run</u>. We will provide feedback only to those students who obtained 0 because of incorrect file names.

Final Due Date
The project is due on **5th Nov, 11:59 PM ET**. Late submissions will NOT be accepted. The deadline is firm, if your timestamp is 12:00 AM, it is a late submission. Please start early. **No regrading requests will be entertained**.

FAQs and Tips:
1. In this project, as you work and play with Solr, you may need to refer to Solr Reference Guide frequently to complete your tasks.

2. For windows, to install TREC on your machine, follow these steps:
   a. install cygwin
   b. unzip trec_eval zip file
   c. move to trec_eval folder in cygwin terminal and execute make, and you will be good to go

3. 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.

4. 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.

5. Should I work with schema or schema-less mode?

      - You can either work with or without schema, the performance won't be different.

6. Which Solr version should I use?

      - You can use any version above 6.0 as you prefer.