CS172: Information Retrieval

UC Riverside

Prof. Mariam Salloum Fall 2019

Final Project

The final project must be done in teams of three. If you cannot find a partner, email me or the TA to connect you to other students who are looking for a partner. We will be using the following Google Doc to keep track of teams and individuals looking for a team: **Team List Google Doc**. Please update once you form your team.

You will prepare a final report outlining the problem you tackled, design of your system, results, and a 'Collaboration Details' section. The"Collaboration Details" section should clearly specify the contributions of each member of the team. I expect your github repo to reflect the contributions outlined in the section, so be sure that everyone contributes to the github repo.

You have a choice of building a search engine for the Web or for Twitter. In Part 1, you will build a crawler (from scratch) to either crawl .edu URLS or collect tweets from the Twitter Streaming API. In both cases, you should collect at least > 1 GB of data. In Part 2, you will use an opensource tool called Lucene `http://lucene.apache.org/core/` to index all HTML files or tweets. You can use ElasticSearch to pose queries on the index created. Finally, in Part 3 you will propose and implement an extension of the search engine created in Parts 1 & 2. Ex. you can build an interactive interface for the user, or you can record query logs and utilize this information for ranking, or personalization.

1 Part 1 - Crawling

1.1 Option 1 - Web Crawling

Your application should read a file of seed .edu URLs and crawl the .edu pages. The application should also input the number of pages to crawl and the number of levels (hops (i.e. hyperlinks) away from the seed URLs). All crawled pages (html files) should be stored in a folder.

We recommend using Java or Python, which is the language that we will use in the discussion sections. If you use another language, you cannot expect to get any support from the TA if you get stuck. You should not use any crawler package, since the purpose of the project is to see some of the challenges involved in building a crawler.

You will be graded on the correctness and efficiency of your crawler (e.g., how does it handle duplicate pages? Or is the crawler multi-threaded?).

1.2 Option 2 - Twitter:

Use the Twitter Streaming API to collect geolocated tweets. Store tweets in one large file, one tweet per row.

If a tweet contains a URL to an html page, get title of that page, and add title as an additional field of the tweet, that is, include it in the JSON of the tweet, so it becomes searchable in Part 2. Hint: Retrieving title in real time may slow down your Twitter stream and lead to lost data, so you have to think about your crawler design.

2 Part 2 - Retrieval

Build index using Lucene. In this part, you will build an index over the crawled data for the purpose of retrieval. Since we build an index and retrieval system for PS1 & PS2, lets use an open-source tool this time around.

2.1 Option 1 - Web

Write a program that uses the Lucene libraries to index all the html files in the folder you created in Part 1. Handle different fields like title, body, creation date (if available).

2.2 Option 2 - Twitter

Write a program that parses the JSON objects of your big files from Part 1 and inserts them into Lucene. Handle the fields like username, location, and so on.

Test your system to ensure that it retrieves relevant documents for a given query. Part of the requirement is understanding how Lucene (ElasticSearch) works, so you should do a Google search to understand the index and how to use it.

3 Part 3 - Extensions

In this Part you have the flexibility to choose a feature of your system to enhance or extend. You can choose an extension from the list below or come up with your own extension idea. Be sure to run the idea by the TA or myself for approval. Ideas:

1. Web-based Interface You can decide to create a Web-based interface. The interface should contain a textbox, and a search button. When you click search, you should see a list of results (e.g., first 10) returned by Lucene for this query and their scores. The list should be ordered in decreasing order of score. Handle fields as you deem appropriate. For Twitter, order by a combination of time and relevance; describe your ranking function. You can use the web development language of your choice. **Do not use SOLR or another framework that automatically builds the UI for you.**

2. Snippet Results Extend system to display a snippet for each result. You can use ideas discussed in class or your own ideas to come up with a good snippet generation algorithm. **Don't use Lucene-generated snippets.**

3. PageRank Alternatively, you can use PageRank to rank pages, that is, you can allow the user to either rank by Lucene ranking or by PageRank.

4. Using Spark Using Spark or MapReduce platform to process large data. Proposal and project report must justify the use of either platform.

5. Twitter geolocations For Twitter projects, you can build a UI that allows for search and display results on map. Read user's location from browser if available and center map there. Only show tweets within 100 miles from user location.

4 Deliverables

The deliverables includes :

1. Checked-in code to github
2. Final report
3. Demo to TA during discussion section

Ensure you have the following items in your final report. For each 'part' include a collaboration details section to describe the contribution of each team member.

4.1 Part 1 - Crawler

1. Collaboration Details: Description of contribution of each team member.
2. Overview of system, including (but not limited to)
   (a) Architecture
   (b) The Crawling or data collection strategy (do you handle duplicate URLs, is your crawler parallel, etc.)
   (c) Data Structures employed
3. Limitations (if any) of the system.
4. Instruction on how to deploy the crawler. Ideally, you should include a crawler.bat (Windows) or crawler.sh (Unix/Linux) executable file that takes as input all necessary parameters. Example instructions for Web-based assignment: [user@server]./crawler.sh $< seed - File : seed.txt >< num - pages : 10000 >< hops - away : 6 >< output - dir >$
5. Screenshots showing the system in action.

4.2 Part 2 - Indexer

1. Collaboration Details: Description of contribution of each team member.
   of system, including (but not limited to):
   (a) Architecture
   (b) Index Structures
   (c) Search Algorithm
2. Limitations of system.

3. Instructions on how to deploy the system. Ideally, you should include an indexer.bat (Windows) or indexer.sh (Unix/Linux) executable file that takes as input all necessary parameters Example: [user@server] ./indexer.sh $< output - dir >$

4.3 Part 3 - Extension

Detailed description of your 'extension' and motivation or benefit of the implemented feature or extension. Include screen shots of your system in action.

4.4 Demo

Furthermore, each team will demonstrate their project for 5 minutes to the TA during the last discussion section. If a team cannot demonstrate at discussion section due to time constraints, you can meet with the TA or myself (by appointment) to demo. Demos will be accepted Week 10 and Finals week (Monday - Thursday).

4.5 README

Include a README file that describes the design of your code and detailed instructions for running your program. We will not grade programs that don't compile, and won't attempt to run your program if the instructions are not provided.

5 Submission

You will be submitting your via Github. Your submission should include all your source files, a README files, and the output data (for the provided data files). The README file should include the names of all group members and a short description of what is included in the submission and how to run your program.

NOTE: its your responsibility to verify that your code was checked-in to the Github repo correctly and ontime.