Course Project

CS 242: Information Retrieval & Web Search

Winter 2018

**Build a Search Engine**

You must work in teams of three. If you cannot find a partner, email the TA (Mohiuddin if you are ground student and Nhat if you are online) to connect you to other students who are looking for a partner. Teams must be formed by end of 2nd week of classes, and their composition emailed to the corresponding TA.

Each project report must have a section called "Collaboration Details" where you should clearly specify the contributions of each member of the team.

**Part A: Collect your data and Index with Lucene**

A1: You have the following options:

a.   Crawl the Web to get Web pages using jsoup (<http://jsoup.org/>). You may restrict pages to some category, e.g., edu pages, or pages with at least five images, etc.

b.   Crawl the Web to get images with their captions and names (to be used for indexing in next parts) using jsoup (<http://jsoup.org/>). Only use smaller imaged (<200KB) so you don’t stress our Hadoop cluster later.

c.   Use Twitter Streaming API (<https://dev.twitter.com/docs/streaming-apis>) to get Tweets.

d. Your own ideas for a dataset are also acceptable, pending instructor approval.

Collect at least 5 GB of data, but no more than 10GB.

We recommend using Java. You may not use a crawling tool.

A2: Index your data using [Lucene](http://lucene.apache.org/core/) (not Solr)

You will be graded on the correctness and efficiency of your solution (e.g., how does the crawler handle duplicate pages? Is the crawler multi-threaded? How do you store the incoming tweets to maximize throughput?), and the design choices made when using Lucene (e.g., did you remove stop words, and why? Or did you index hashtags separately from keywords and why?).

Deliverables:

1: Report (5-10 pages) in pdf that includes:

* 1. Collaboration Details: Description of contribution of each team member.
  2. Overview of the crawling system, including (but not limited to).
     1. Architecture.
     2. The Crawling Strategy.
  3. Overview of the Lucene indexing strategy, including (but not limited to).
     1. Fields in the Lucene index, with justification (e.g., indexing hash tags separately due to their special meaning in Twitter).
     2. Text analyzer choices, with justification (e.g., removing stop words from web documents; using separate analyzers for hashtags and keywords).
     3. Report the run time of the Lucene index creation process. E.g., a graph with run time on the y axis and number of documents on the x axis.
  4. Limitations (if any) of system.
  5. Obstacles and solutions
  6. 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:* [user@server]./crawler.sh <seed-File:seed.txt> <num-pages: 10000> <hops-away: 6>  <output-dir>
  7. Instruction on how to build the Lucene index.
* Ideally you should include an executable file that takes as input all necessary parameters. *Example:* [user@serer] ./indexbuilder.sh <input\_dir> <Analyzer Options>

2: Zip file with your code

Submit your report and zip file through iLearn by **2/9**.

**Part B: Index your data using Hadoop and build Web Interface**

B1: The hadoop program should input the collected data (which should be stored in large files (>1MB) to avoid crashing Hadoop) and output an inverted index, where for each word we store at a minimum its locations and frequencies in the documents. JSON is the recommended format for input or output files. You may use a NoSQL database system, e.g., Cassandra or HBase, to store your index and support fast queries, or just carefully organized files.

B2: Build a Web interface to search your data

At the minimum you should have a search textbox and a way to select between your hadoop or your lucene index.

You will be graded on the correctness and efficiency of your MapReduce jobs and the comparison of the runtime between Hadoop and Lucene index construction; you will also be graded on the design choices made with respect to the indexes created by Hadoop (e.g., how do you split the Hadoop-generated index to files and how they are read by the Web application) and the methods used to rank documents for both Lucene and Hadoop.

We recommend Angular JS on Tomcat, but you are free to use your own Web-based programming language (at your own risk).

***Extra credit:*** 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.

You will be graded on the correctness of you solutions, their efficiency, the insightful discussion in your report, the novelty (interestingness) of your application, and the presentation.

 Deliverables:

1: Report (5-10 pages) in pdf that includes:

1. Collaboration Details: Description of contribution of each team member.
2. Overview of system, including (but not limited to):
3. Architecture
4. Details of how hadoop was used (key, values definitions and data flow).
5. Explain and justify the indexes built by hadoop. E.g., did you stem the keywords and why? How did you handle special fields such as hashtags?
6. Explain how you use the hadoop-created index to do the ranking
7. Explain how you use the Lucene index to return results. E.g., what options did you use with the QueryParser? How were multiple fields in the Lucene index used with the QueryParser?
8. Report the run time of the Hadoop index construction. This should be comparable with the report on the run time of the Lucene index construction from report A. There should also be a discussion explaining the differences between the run times.
9. Limitations of system.
10. Obstacles and solutions
11. Instructions on how to deploy the system.
12. Ideally, you should include an indexer.bat (Windows) or indexer.sh (Unix/Linux) executable file that takes as input all necessary parameters e.g.
    1. **Example:** [user@server] ./indexer.sh <input-dir> <output-dir>
13. Create a web-application (e.g. Web Archive) that can be deployed to a webserver like Tomcat.
14. Screenshots showing the system in action.

2: Zip file with your code

Submit your report and zip file through iLearn by **3/16**.

Further, each ground team will demonstrate their project in the class for 5 minutes at a specified time.

Each online team will prepare a 5-min presentation video and include the link in their report. All members of each team should participate in the presentation. E.g., you could each present a part of the project and concatenate the presentation videos into a single video file.