Semester Project Progress Report

By: Christen Ford

ID: 2741896

Platform Setup and Tools Employed

The platform for this project has the following specifications:

* Ryzen 2700u @ 3.8Ghz
* 8GB 2133Mhz DDR4 Ram
* 512GB NVMe Host Drive
* 1TB 7200RPM External USB3 Data Drive

The data files for this project are stored on the data drive listed above. This drive is externally connected to my computer through a USB3 port and serves as a bottleneck for the project.

In terms of software, I am employing the following for this project:

* Python 3.7.2
* Snowball (Python stemmer) 1.2.1
* MongoDB 4.0.6
* NodeJS 8.12.0
* Express 4.16.4
* Pug 2.0.3
* Snowball (NodeJS stemmer) 0.3.1

Python is used for parsing the raw eBook files and uploading the metadata to MongoDB. Python is also used to generate intermediary stem files, parse them, and then upload them to MongoDB (for the search process).

Data Collection

It turns out that bulk downloading eBooks from Project Gutenberg is rather difficult to do. I would either must become a mirror of the project or very patiently wait for a whole bunch of files to download that includes a whole bunch of miscellaneous files that I do not need (wasting my metered bandwidth at home).

To compromise, I have downloaded an ISO containing a 2010 backup of the library, which has roughly ~10,000 eBooks in it. I had to filter these eBooks before making them part of my dataset as some of them were just the same eBook in a different encoding, or some were eBooks that were primarily composed non-text-based information (meaning they would be useless for my purposes).

My final dataset has roughly ~8,600 eBooks in it for a total of ~2GB of data.

Data Preprocessing/Transformation

I already have the entire dataset parsed and uploaded as two collections within a MongoDB database. The first collection stores metadata on each eBook such as the title, author, and file path of the eBook on disk. The second collection stores the stemmed text content of each eBook (for querying in the search process). Both collections are indexed via their unique book id for faster lookups.

The file format for the eBooks from Project Gutenberg have been quite a pain to work with. They are just plain text files, but each is organized into three sections: a header, content, and a trailer. The header contains all the metadata for the eBook and was very difficult to process (more discussed in the problems section below). The content section contains the actual text of the eBook and is what I parsed to generate each stem file. The trailer contains the Project Gutenberg license and terms of use. it was not parsed as it did not contribute any useful information to the project.

Problems Encountered and Resolutions Enacted

It seems that running MongoDB on a single node and invoking aggregation on it for many documents without using the $match operator to limit the query is very slow. The issue is that I need to do it, there’s not really a way around it, short of using HDFS and Hive or Pig.

I’d prefer not to as my laptop is unreliable in Linux. That is, the system will randomly lock up (forcing me to hard reboot). This occurs both when Linux is installed natively as well as when I am running it inside of a virtual machine. While MongoDB is cross platform, HDFS and its suite of programs are not.

I was able to make them work for the lab, but I have already encountered issues with trying to use them on the previously mentioned machine for this project. The result corrupted my entire MongoDB database, causing me to not only lose all of the information for this project, but all of the information for each lab previously completed for this course.

As stated in the prior section, I also had problems parsing the header of each eBook. The header tends to follow a standard format, however there are a good number of eBooks in the library that differ their header from the standard format. The header has important metadata prefixed with a predictable key value paired with the corresponding content. For the most part, these keys are the same, however there were sometimes typos (which the parser accounts for).

The major issue (which I do not account for) is that sometimes the corresponding value for the key was split across multiple lines in the file. This results in certain keys having incomplete information corresponding to them in the dataset, however this is inconsequential. The metadata is only useful as display on the search results. The data used by the search engine is the stemmed text content of the eBook (which parses fine). I should be note that I tried to fix this issue in many ways, however I was unable to get it working and for the sake of time, I set it aside to work on other parts of the project.

Remaining Work

As of right now, I am currently trying to get the indexing working. Specifically, I am trying to generate either a MapReduce or Aggregation pipeline for MongoDB that returns te relevant search results to the Express/Pug frontend. Speaking on that, the frontend is nearly complete, however I still need to complete pagination on the search page so the user can page through the results returned by MongoDB.

When completed, MongoDB should send back the top 100 relevant results to the user. These results will then be displayed in a paginated format where each result lists the eBook name, author and filelink. Clicking on the link should allows the user to view the eBook. The most important part of this project is ensuring the results returned by MongoDB are actually relevant to the users query (which entails a lot of testing on my part).