Mini Project 2

A design document submitted in partial fulfilment of
CMPUT291

November 24th, 2019

Raynard Dizon
Daniel Florendo
Ibrahim Aly

System Overview

This project functions as a database for email records. An xml file contains various records of emails containing the subject, body, sender, recipient, etc., for each record and users of the program are able to look up these records using certain fields. The program is divided into three phases. In phase 1, the xml is parsed into several .txt files namely terms, recs, emails, and dates. Next, in phase 2 these .txt files are converted into .idx files. In phase 3, user input is collected, parsed, and used to query the files from phase 2. The program returns appropriate email records based on the criteria provided by the user.

To run the program, all three phases must be executed. Run `phase1.py` located in the *phase1* directory by executing the code below in a terminal:

```
>>python3 phase1.py 1k.xml or  >>make 1k
```

Once the required .txt files are generated, navigate to the root directory and start phase2 using:

```
>>./phase2.sh
```

The final step is to run phase 3 and start querying. Run the program by executing:

```
>>python3 main.py
```

An example query that returns all emails with ‘gas’ in its subject field is: ‘subj:gas’.

Algorithm

In the first part of the algorithm we parse in input from STDIN. To do this, we first check if the expression is a valid expression using regex. Upon knowing the input is valid, we then check to see if it is a query or a mode change. If it is a mode change, the function will return what to change the mode change. If it is a query then each part of the query is split. Now, having the query split, we parse each input according to what type of query it is. An array of arrays is returned. The array contains arrays corresponding to each of the queries passed. Each array within the array contains information on what the query was asking. In the querying algorithm, the program loops through each query parameter (the list created), finds out which database to query, and performs the given query. For querying emails, it finds out all the duplicate keys that are equal to the given email. For querying terms, if there’s no ‘s-’ or ‘b-’ in the beginning, it searches for the given term with ‘s-’ or ‘b-’ at the beginning. If there is a ‘%’ at the end, it finds all the keys that start with the given term. Otherwise, it finds all the duplicates keys that are equal to the given term. For querying dates, it queries based on the given date. Each query will return a

set. Then, the program will intersect all those sets to find the Row IDs that are common between all the queries. Finally, the program will display the details regarding each found Row ID.

Testing Strategy

To test the parsing portion of the program, inputs were first tested in regex101.com to verify that the pattern was correctly identifying input according to the grammar specified by the project. Once patterns were verified to be working correctly, sample data from eClass were used to test the querying portion of the program. The full and brief functionality was checked last.

Group Work Strategy

The project break-down between partners, including time spent for each task, is as follows:
Raynard Dizon:

- Phase 2 (1 day)
- Phase 3: Parsing the query (1 day)
- Testing (1 day)

Daniel Florendo:

- Phase 1 (1 day)
- Phase 3: Displaying the output in full or brief format (1 day)
- Testing (1 day)

Ibrahim Aly:

- Phase 2 (1 day)
- Phase 3: Querying (1 day)
- Testing (1 day)

GitHub was used to facilitate remote development. Scrums were done in Messenger and Discord.