Mini Project 2 Design Document

ridderik | yuhang5 | wstix

OVERVIEW

This project is a database retrieval system implemented in Berkeley DB. It's split up into three stages: data retrieval, data sorting, and data querying, in that order. Each time a new database is to be opened from an xml file containing email data in the specific format specified by the project, the first two steps must be completed for the third to be possible.

USER GUIDE

The user starts with an unsorted xml file containing email data. To access it using our interface, they first extract the data in Phase 1 by running phase1.py and entering the email data's filename. This sorts the data into text files. Next, the user should run phase2.py, which turns the extracted data into the databases to be queried. Finally, the user can launch menu.py, and use the respective keyword for each email field, followed by an email or a word, to find matching emails and words in those fields. The keywords are as follows: `to:`, `from:`, `cc:`, `bcc:`, `subj:`, `body:`, `date : or < or > or <= or >=`, or a word by itself (this will query both the subjects and the bodies of the emails). The program will return the email's row id and the email's subject by default, but with output=full, the whole record is shown.

QUERY ALGORITHM

Our algorithm mainly involved using the DB_SET_RANGE flag to perform range searches on the B+-Tree index files, making use of python's `in` operator when performing partial match searches. In every case, the intersection of all entered queries is taken, with the main presupposition in the algorithm being that

each email can only have one sender. Set operations of union and intersection in python were used to combine the lists of row ids that came out of each query on terms and emails. These row ids were then used to retrieve the records from the re.idx file.

TESTING STRATEGY

We tested this project both individually and together, breaking the program down into smaller parts and testing individual queries and databases to compare them to the expected results from eClass. In particular, we tested for case-insensitivity, multiple queries, algorithm performance, database correctness, and incorrect queries. We feel that the project has been thoroughly tested, and should not cause too many issues.

GROUP WORK STRATEGY

First, we delegated the first two phases and the writing of this report among our group members. The entire first phase was dealt with by Cameron, the entire second phase by Yuhang, and this report was written by Warren. We decided to break the third phase of the project up into story points and assigned an equal number of story points to each group member. The breakdown is as follows (sp means Story Points):

• Retrieve a query from the user and convert it to something usable by the rest of the program: 2sp (Cameron), queries for emails: 2sp (Yuhang), date queries: 2sp (Yuhang), body and subject (term) queries: 2sp (Cameron), query ordering: 4sp (Warren)

Each group member spent around five hours working on the project individually at home, and we spent around 20 hours working on the project together in person. We communicated through Slack, and maintained the project in a GitHub repository, making pulls and pushes as needed.