Project Proposal: A Portable Large Volume Email Retrieval System

Shivaan Motilal
University of Cape Town
Computer Science
mtlshi005@myuct.ac.za

Breyden Monyemoratho
University of Cape Town
Computer Science
mnybre002@myuct.ac.za

CCS CONCEPTS

• **Information systems** → **Information retrieval** → Specialized information retrieval • **Software and its engineering** → Software libraries and repositories • **Software and its engineering** → Software Portability

KEYWORDS

Portable; Searchable; Email formats; Archives; Indexing; Parsing; User interface; Query System

1. PROJECT DESCRIPTION

In the working world, there is often a need to search and browse through very large collections of emails, to track down individuals or to verify decisions, etc. For convenience, many users will either delete or archive email after it has been handled. If they choose to archive their email, these archives can later become large and cumbersome to search through, especially after a long period of time has passed. This can be attributed to many factors, including poor personal information management and large amounts of high priority email. Whittaker et al. [15] termed this "email overload".

With the problem of "email overload", there is also the issue of archives becoming obsolete through *software aging* [8]. In order to combat obsolescence and improve longevity, various preservation strategies need to be considered.

A possible solution to address the email overload and obsolescence issues, is to use a portable offline searchable email archive which handles multiple email formats (such as mbox and maildir). The searchability feature would allow for specific emails to be retrieved from the large archive (managing email overload), whilst the portable and offline features would make the archive less likely to become obsolete in the short-term. This solution is the one proposed in this paper.

Taking the above into consideration, we will be creating a Web application to facilitate portability and offline searchability, that allows for multiple email formats as inputs.

The project is divided into two logical sections which when used in conjunction, will solve the overall project. The two sections are as follows:

1. *Pre-Processing:*

   This involves parsing and indexing of the inputted archives of various email formats. Parsing will extract and structure relevant information from the inputted archive, whilst indexing will involve creating indices from the parser output.

2. *Email Processing:*

   This will be composed of creating the user interface and a query system that allows for fast and efficient retrieval of emails. The user interface should display emails clearly to the user and allow for ease-of-use. The query system should be able to handle various queries.

2. PROBLEM STATEMENT

Our proposed solution will consist of the following features:

1. *Should allow for input of large email archives in multiple email formats:* large maildir and mbox archives(as a basic requirement) should be processable by the Web application.

2. *Be Portable:* Due to a Web application being used, portability is already somewhat achieved as the email system should be able to run on multiple platforms that support a Web browser.

3. *Provide offline usage of the archive/collection*: features such as search and browse should be available without Internet connectivity.

The research questions are:

2.1. Pre-processing: Indexing and Parsing

- Can we create an indexing system that works for the popular email formats, as well as other relevant ones?
- Can both indexing and parsing work on multiple platforms (portable)?

2.2. User Interface and Querying System

- Is it possible to create a user interface that represents emails in a easy to understand way, and is usable?
- Can a query system be created that allows for fast and accurate retrieval of email?

3. PROCEDURES AND METHODS

Our approach will be split into the two main sections of pre-processing and email processing.

3.1. Pre-processing

3.1.1. Parser

Before indexing can occur, email archives (sources/inputs) of different formats, need to be streamed into the application. The parser will extract relevant data (constituents) from the email archives and pass this data on to the indexer. We plan to create our parser in the Python programming language, using the existing mailbox module [9].

3.1.2. Indexer

The indexer will then create metadata, search terms and apply tags from the output of the parser, creating multiple indices from the information. There will be two indexers, one for the search functionality and the other for browsing. The indexers will also be created using Python.

3.1.3. Evaluation:

The following evaluation criteria will be used to test the parsing and indexing components:

- *Portability* - check whether the system functions on multiple browsers (Firefox, Google Chrome, Safari, Microsoft Edge etc.)
- *Efficiency* - test the time taken (speed) to parse or index.
- *Effectiveness* - for the parser, the number of correct constituents in the parser output will be measured. Parsed constituents comprises of data from email attachments, data fields from the email body and email headers. For the indexer, the time spent searching and browsing compared to looking through the larger archive, will be measured.

3.2. Email Processing

3.2.1 User Interface(UI)

The user interface will consist of mainly static HTML and JavaScript to display the emails from the archives clearly to the user. For the browsing functionality, each email will be its own static HTML page, and linked to a main page that shows the repository in a structured layout. To develop the UI, we will use a user centered approach and consult users in order to understand users' needs and preferences.

3.2.2 Query

The query system will retrieve relevant email from the email archive, by using the indices generated by the search indexer.

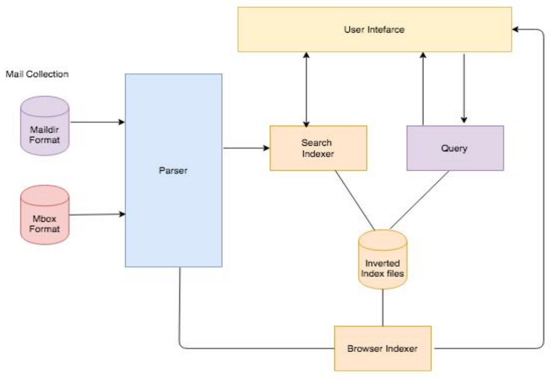


Fig. 1. Overview of the FINDMAIL system

In Fig.1, we show the popular email formats: maildir and mbox, being inputted to the parser. The parser then sends its output to the two indexers. The browser indexer will create indices to facilitate browsing of the email, whilst the search indexer will create indices for the search functionality. Both of these indexers will interact will the user interface to provide the services of browsing and searching to the user.

3.2.3. Evaluation:

The following evaluation criteria will be used to test the UI and query components:

- *Usability* - is concerned with attributes of the system that make it understandable, learnable, easy-to-use and attractive. Jakob Nielsen's heuristics [3] will be used to assess the usability.
- *Efficiency* - test the time taken (speed) to browse or search and generate the results view within a reasonable time (for the size of the archive).
- *Effectiveness* - measures the relevance of the search results. We will measure this by obtaining the percentage of the relevant items retrieved from the total number of emails.

3.3. Resources

We plan to use Python, HTML and JavaScript to create the components of the FINDMAIL system. The decision to use the Python programming language, is mainly motivated by the need for digital preservation through simplicity. Apart from being easier to understand, Python is very portable and can run on many operating systems. Python also has packages and modules which can be used for indexing and parsing email archives.

4. ETHICAL, PROFESSIONAL AND LEGAL ISSUES

Ethical issues have been identified in the testing, software implementation and data handling stages of the project. Each will be discussed in further detail below.

4.1. Testing:

We will apply to the Faculty of Science Research Ethics Committee for ethical clearance, in order to test the usability of the system on students and staff.

4.2. Software:

This project will be declared open source. This is to encourage further development and improvement to our software.

4.3. Data:

The data to be used on the system will be sourced from our own personal Gmail inboxes (Shivaan Motilal has a 680 MB inbox unzipped, Breyden Monyemoratho has a 670 MB inbox unzipped) and compiled from the Enron email dataset containing approximately 1.5 million emails (423MB, tarred and gzipped) [2]. We have an ethical responsibility to take into consideration the privacy of the parties involved, particularly in the case of the Enron email dataset.

5. RELATED WORK

5.1. Digital Collections:

For developed countries many preservation techniques can be implemented, however this is not the case for developing countries (such as in Africa) [12]. In developing countries, most preservation techniques cannot be implemented due to insufficient resources and poor/expensive cost of Internet bandwidth.

A particular way of preserving digital collections(including email archives) that works for developing countries is through using the principle of simplicity [8]. An illustration of this could be the use of XML plain text documents to store information and metadata, making it more likely for the information to be retrievable in later years. Focusing on simplicity also provides easier interconnection, extension and modification of the features of a specific system, allowing for the system to function on multiple platforms(portable). The concept of portability is important for email, as email users use multiple platforms to access their email, and the email itself can be stored in different formats.

Suleman et al. [13] developed CALJAX, a generic hybrid (online-offline) repository management and access system based on a strong AJAX foundation. It allows integration of content from a local source with content from a remote source, with the only requirement being a Web browser. XML plain text documents were used to store information, making it more likely for the information to be portable, preservable and accessible through a Web browser.

Expanding on the issue of poor Internet bandwidth, is the idea of having hybrid online-offline digital collections to counteract this issue. Online and offline collections present both advantages and disadvantages, thus a hybrid digital collection(online-offline repository) could interleave advantages from both, and potentially aid in preservation [13]. A hybrid system will however not be in the scope of this project, as there are many complications that come with creating such a system.

5.2. Email Archives:

Some existing software projects around email archives include Windows Mbox Viewer(WMV), Mairix [10] and Mailpile [6]. WMV [11] displays mbox files on the user's screen via a simple user interface. It runs offline but is a program specifically for Windows. It also does not provide search functionality over the archive. The other downsides are the fact that it does not cater for other email formats and is not portable across operating systems.

Mairix and Mailpile include indexing and search functionality, but are not suitable either in terms of preservation, portability or offline use. Mairix [10] is an email indexing and searching tool that works with maildir, MH or mbox formats. It works offline but is mainly for Linux systems. Since it involves installation and is not portable across non-Linux operating systems, it is unusable in this project.

Mailpile [6] is similar to Mairix; it also indexes mbox and maildir formats, however Mailpile is an email client and personal Web mail server. It also has a much better user interface(in comparison to WMV) that is based on Gmail. It works on multiple browsers but does not have specific offline usability. It was made using Python, JS and HTML5, and is the closest work to the one we propose in this paper.

6. ANTICIPATED OUTCOMES

The project aims to develop a robust, portable and large volume email retrieval system for an individual, that can be used both online and offline.

6.1. Key Features

- Portability (function on multiple platforms).
- Provide offline access.
- Fast and accurate search and browse functions over email (handle various queries).
- Intuitive user interface.
- Parsing and indexing of popular email formats (mbox and maildir).

6.2. Major Design Challenges

- Making the indexer and parser work with multiple email formats.
- Module integration of the indexer and parser, with the user interface and query system.
- Module integration (UI, Parser, Indexer, Query System).
- Designing the system such that it works for multiple Web browsers.

6.3. Expected Impact

We hope this project will help individuals better manage their emails from large archives and provide them with fast and accurate search and browse functions, making it more likely for the information to be retrievable in later years. In addition, by releasing the resulting tool from this project as open source, the community will be able to freely use our work and expand and improve on our tools.

6.4. Key Success Factors

- Our system is able to parse and index multiple email formats.
- Provides offline access.
- Provides fast and accurate search and browse functions over email.
- Considered portable (function on multiple platforms).

7. PROJECT PLAN

7.1. Risk Matrix

The risk matrix in Appendix A outlines the probabilities and impact of each risk identified. It also outlines the consequences, mitigation strategies, monitoring strategies and management strategies that will be applied to each risk.

7.2. Timeline

A timeline of events and deliverables are shown in the Gantt Chart in Appendix B.

7.3. Resources Required

7.3.1. Software required

We anticipate we will need to make use of the following software resources:

- *Visual Studio Code*- This is an IDE that supports Python, HTML and JavaScript code editing (along with other programming languages). It is a well-established software and has multiple features for coding, debugging and compiling efficiently [14].
- *The python mailbox module*: This module allows for manipulation of emails in various formats including maildir, mbox, MH, Babyl and MMDF [9].
- *Jest*- This will be used for JavaScript testing and is a popular Javascript testing tool created by Facebook [6].

- *JSDOM*- Allows for testing of JavaScript in a simulated browser environment [6]. This will be used before testing is done on real browsers.

7.4. Project milestones

Table 1: Table showing the project milestones

Date	Description
25-05-18	Project Proposal Presentation
11-06-18	Revised Proposal Submission
15-06-18	Ethical Clearance Request Submission
15-06-18	Project Web Presence
23-07-18	Initial Software Feasibility Demonstration
25-08-18	First Draft of Final Paper
27-08-18	Final Complete Draft of Final Paper
06-09-18	Project Paper Final Submission
07-09-18	Project Code Final Submission
17-09-18	Final Project Demonstration
17-09-18	Project Poster
26-09-18	Project Web Page
03-10-18	Reflection Paper

7.5. Software milestones

Key software milestones comprises of the back end, front end and services modules.

- Back end module - Consists of the indexing and parsing components of the application.
- Front end (interfaces to the services) - User interfaces.
- Services (Service functionality) module - search and browse services.

7.6. Work Allocation

Shivaan Motilal will be working on the pre-processing components, namely the parser and indexer. Breyden Monyemoratho will be working on the user interface and query system for email processing. The format of the indices will be determined and worked on together.

8. REFERENCES

[1] CALO Project. Enron Email Dataset, 2015.DOI: https://www.cs.cmu.edu/~enron/

[2] Centre for Curating the Archive. The Digital Bleek and Lloyd, 2018. DOI: http://lloydbleekcollection.cs.uct.ac.za/

[3] Jakob Nielsen and Rolf Molich. 1990. Heuristic evaluation of user interfaces. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90), Jane Carrasco Chew and John Whiteside (Eds.). ACM, New York, NY, USA, 249-256. DOI=http://dx.doi.org/10.1145/97243.97281

[4] JSDOM. Javascript browser simulator, 2018. DOI: https://github.com/sttk/jsdom-browser/

[5] Facebook. Jest. Javascript testing tool, 2018. DOI: https://facebook.github.io/jest/

[6] Mailpile. An email client, 2018. DOI: https://www.mailpile.is/

[7] David L. Parnas. Software aging. In *Software Engineering, 1994. Proceedings. ICSE-16., 16th International Conference on* (pp. 279-287). IEEE. May, 1994.

[8] Lighton Phiri and Hussein Suleman. In search of simplicity: Redesigning the digital bleek and lloyd. *DESIDOC Journal of Library & Information Technology,* (pp 32-34), 2012.

[9] Python 3 Standard Library. Mailbox module, 2018. DOI: https://docs.python.org/3/library/mailbox.html/

[10] SourceForge. Mairix. Programme for indexing and searching mail, 2009. DOI: https://github.com/rc0/mairix/

[11] SourceForge. Windows Mbox Viewer, 2015. DOI: https://sourceforge.net/projects/mbox-viewer/

[12] Hussein Suleman. An African Perspective on Digital Preservation. In *Multimedia Information Extraction And Digital Heritage Preservation* (pp. 295-306), 2008.

[13] Hussein Suleman, Marc Bowes, Matthew Hirst, and Suraj Subrun. Hybrid online-offline digital collections. In *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on - SAICSIT '10*, ACM Press, 421-425, 2010.

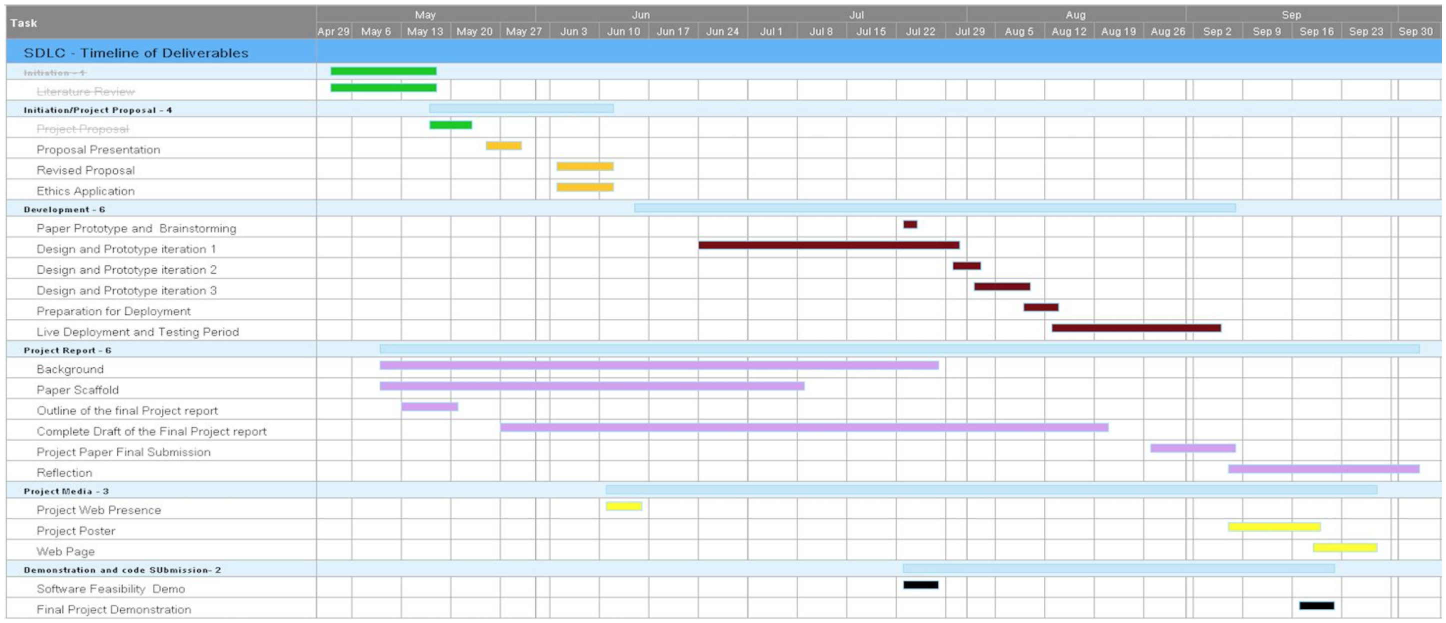
[14] Visual Studio Code. A compact code editor and IDE, 2018. DOI: https://code.visualstudio.com/

[15] Steve Whittaker, and Candace L. Sidner. Email overload: exploring personal information management of email. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 276-283). ACM. April, 1996.

Appendix A: Project Risk Matrix

Risk	**Probabi lity**	**Impact**	**Consequence**	**Mitigation**	**Monitoring**	**Management**
Scope creep.	High	High	Change in requirements and more pressing time constraints.	Allocate time buffer to cater for unexpected changes/complete project within the timeframe.	Keep up with the changing needs of the stakeholders.	Adhere to an agile approach to the project.
Members do not complete their section on time.	High	Medium	Work not being delivered in time and increase in pressure to complete project.	Have sections split such that each can work as standalone parts or use dummy data as input to the particular module that has the dependency, to test functionality.	Keep tabs on teammates, plan events.	Have discussion amongst team members and facilitator, to determine which sections can be prioritized over others.
Inability to meet the project sub-deadlines.	Medium	High	Loss of marks.	Do additional planning and have more finer-grain/specific deadlines.	Work to make up delay or remove excess functionality tasks/prioritize.	Have discussion amongst team members and facilitator, to prioritize which sections to work on.
A team member drops out of the course.	Medium	High	Work not being delivered in time and increase in pressure to complete project.	Maintain a consistent level of contact with team members, have sections split such that each can be standalone sub-projects.	Keep tabs on teammates, plan events and encourage clear and honest communication.	Split tasks amongst other team members, communicate problem to su pervisor.

Appendix B: Gantt Chart


Task
May
Jun
Jul
Aug
Sep
Apr 29
May 6
May 13
May 20
May 27
Jun 3
Jun 10
Jun 17
Jun 24
Jul 1
Jul 8
Jul 15
Jul 22
Jul 29
Aug 5
Aug 12
Aug 19
Aug 26
Sep 2
Sep 9
Sep 16
Sep 23
Sep 30
SDLC - Timeline of Deliverables
~~Initiation - 1~~
~~Literature Review~~
Initiation/Project Proposal - 4
~~Project Proposal~~
Proposal Presentation
Revised Proposal
Ethics Application
Development - 6
Paper Prototype and Brainstorming
Design and Prototype iteration 1
Design and Prototype iteration 2
Design and Prototype iteration 3
Preparation for Deployment
Live Deployment and Testing Period
Project Report - 6
Background
Paper Scaffold
Outline of the final Project report
Complete Draft of the Final Project report
Project Paper Final Submission
Reflection
Project Media - 3
Project Web Presence
Project Poster
Web Page
Demonstration and code SUbmission- 2
Software Feasibility Demo
Final Project Demonstration