Master's thesis

Prisma: A prototype for private and offline searching in mbox files

Erik Schill

Examiners:	Prof. Dr. Hannah Bast
	Prof. Dr. Christian Schindelhauer
Adviser:	Prof. Dr. Hannah Bast

Albert Ludwig University of Freiburg
Faculty of Engineering
Department of Computer Science
Chair of Algorithms and Data Structures

August 2nd, 2024

Writing period

02.02.2024 - 02.08.2024

Examiner

Prof. Dr. Hannah Bast

Second examiner

Prof. Dr. Christian Schindelhauer

Adviser

Prof. Dr. Hannah Bast

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.

I also hereby declare that my thesis has not been prepared for another examination or assignment, either in its entirety or excerpts thereof.

Place, date	Signature

Acknowledgements

I would like to thank...

- Prof. Dr. Hannah Bast for being my adviser and the first examiner.
- Prof. Dr. Christian Schindelhauer for being the second examiner.
- the study participants for their time and effort.
- my family for their kindness, encouragement and support.

Abstract

Although email is an open communication protocol, most email users are locked within the system of a single email provider. In the past it was generally impossible to export email data or move the data to a different provider. This changed in 2019 when the GDPR came into force in Europe. The GDPR grants European citizens the right to export their data from online platforms, and archive it privately offline. Users that export their emails offline need simple to use, open-source, user configurable programs to manage their emails without being restricted to a single provider, while still allowing users to send emails as they choose, using an address and service of their choice. In this thesis, we present Prisma, an open source Thunderbird add-on that leverages the CompleteSearch engine to search in mbox files. We conduct a small user study under Thunderbird users to compare the usability of Prisma and the Thunderbird Global Search and Quick Filters. The results are mixed, but give clear directions to further improve Prisma's functionality and usability.

Contents

List of tables

1 Introduction

In this chapter we describe the challenge of managing exported, locally stored emails. We also provide a short description of the different formats in which emails can be stored. Lastly, we mention our work's contribution.

1.1 Managing exported emails

Most email users depend on a single email provider. This means the user accesses their email only through this provider and do not make use of the open protocol underlying email. In the past, it was often difficult or impossible to easily export one's own emails from a provider's platform.

This changed in 2019, when the General Data Protection Regulation (GDPR) came into force in the European Union (EU). The GDPR grants Europeans the right to protect their data and personal information online. It prohibits companies from using or selling the private data of Europeans without explicit consent from the user. Additionally, users have the right to export their data from online platforms, move their data between providers, and delete their data from services [7].

The GDPR enables European users to export emails from a provider's platform and archive the emails offline. However, archiving emails is often not attractive to users, because the users still need to access their emails. Non-technical users are not able to easily access and read their archived offline emails. Accessing archived emails can be necessary in order to search in emails for specific information or to reply to an archived email with another email account.

1.2 Email message formats

Email messages use the Internet Message Format (IMF). The Internet Engineering Task Force (IETF) manages the standards regarding email messages. These standards are published in the form of a Request For Commment (RFC) document. However, not all RFCs are standard. This can cause incompatibilites between different email programs and email providers.

Emails can be stored in mbox (short for mailbox) or maildir (short for mail directory) format. An mbox file stores a number of emails in a single file, one after the other. The mbox file must be locked when a process is reading it. Otherwise another process could edit the mbox file mid-read, leading to the reading process seeing corrupted data.

In case of maildir, each email is stored in a seperate file, which can be organized in a directory structure. Since each email is a separate file, each email can be locked separately. The downside is that it takes more space on the filesystem.

In short, an mbox file is simpler, but it requires a global lock. Maildir takes more space, but allows per file locking.

1.3 The mbox file format

The mbox file format is used to store a collection of internet messages (emails) in a single file. Messages are separated from each other by separator lines. A separator line must start with "From ". A basic message that is ASCII-encoded can be stored directly in a human-readable format. Below is an example of an email message:

```
From server@ip Fri Sep 06 09:54:33 2019
From: some.user@mail.address.tld
To: other.user@mail.server.tld
Subject: Very important message

Hi other user,
This message is very important.
Best,
Some user
```

However, parsing real-world emails is not trivial. There are multiple edge cases. There are many outdated character sets that will be encountered. To complicate matters, mbox is also not a file family. There exist different variants such as mboxo, mboxrd, mboxcl and mboxcl2different. These variants are not compatible with each other.

1.4 Open source email clients that support mbox files

Open source programs exist that can import mbox files and allow the users to interact with the email messages they contain. We will briefly cover two such programs: Mutt[1] and Thunderbird[2].

Mutt is a command line email program for Linux[3]. Mutt can directly open an MBOX file containing emails:

```
mutt -f <path-to-mbox-file>
```

Mutt can search in messages, display messages, and reply to messages or forward them. This is a good solution for users that are already comfortable with using the command line on Linux. However, most email users are not familiar with Linux or with using a command line. A program such as Mutt would not appeal to them. The learning curve would be steep.

Thunderbird is an open source email client program developed by Mozilla and has a graphical user interface. Thunderbird stores emails in the mbox format, and external mbox files can be imported into Thunderbird. Thunderbird also supports add-ons. Add-ons can supplement the Thunderbird program with additional functionality.

1.5 Contribution

In this work we present Prisma, a tool for privately searching in mbox files. We also developed a Thunderbird add-on that can be used together with Prisma. It provides a graphical search interface that allows Prisma results to be opened and sent from Thunderbird.

Prisma consists of four components:

[1]`http://www.mutt.org/`

[2]`https://www.thunderbird.net/`

[3]`https://github.com/torvalds/linux`

1. A Docker container that isolates the tool from the rest of the user's host system. The isolation ensures that none of the user's files are accessible to the tool, unless the user explicitly gives the tool access to a file. Additionally, it ensures that the tool does not in any way modify the user's host system. The tool cannot accidentily create or delete a file or change a file's content.

2. An mbox parser that can parse mbox files into other file formats. So far, the parser is able to output CSV (Comma Seperated Values) and Turtle[4]. The mbox parser runs inside the Docker container.

3. The CompleteSearch search engine [5] that, given a query request, will return relevant emails to the user. CompleteSearch runs inside the Docker container.

4. A Thunderbird[5] add-on that contains the CompleteSearch web interface. The web interface allows a user to formulate queries and view search results returned by CompleteSearch. The Thunderbird add-on runs in Thunderbird on the host system. Setting up the add-on requires some files to be written to the user's host system. Otherwise, the add-on cannot communicate with the Docker container. The script that writes these files prints all the changes it makes to the user. The tool can be used without the add-on, but then it must be used from the command line using a tool such as CuRL[6].

Thunderbird has its own own functionality to search for emails. We conducted a four-person user study to compare Prisma to Thunderbird's own search functionality.

[4]`https://www.w3.org/TR/turtle/#sec-intro`
[5]`https://www.thunderbird.net`
[6]`https://curl.se/`

2 Related work

In this chapter we mention the related work. The first section covers other software implementations that are similar to Prisma in certain regards. The second section covers the related user studies that have been conducted.

2.1 Related software for email search

Michel et al. [12] implement an email search tool inside a Thunderbird add-on. The search tool extracts extra information from the headers of email messages stored in Thunderbird, and include this information in the search results. For example, the tool can show the total number of messages sent by senders of messages that appear in the search results, or the date at which senders had sent their first email.

Rutter [14] presents a Thunderbird add-on designed around the concept of associative memory for email refinding, where the user remembers something about a related email which can help the user find the email they are currently searching for.

2.2 Related studies

Whittaker et al. [17] compare opportunistic methods for finding emails, such as text search, and preparatory methods for organizing emails, such as subdividing emails into folders. They find that searching leads to more succesful retrieval, but users often scroll through the search results. They conclude that a hybrid approach might be optimal.

Qingyao et al. [2] report various differences between web search and email search. They note that in case of email search, users don't only search for information, but also search as part of organizing their emails. Email search queries are also more specific than web queries, and repeat queries for the same results are less likely.

Mackenzie et al. [11] suggest that a hybrid approach of time-based and relevance-based results ranking might be optimal, and observed that users append words to queries or reformulate their entire query.

Szóstek [16] investigates email management according to six needs for email organization and three needs for email retrieval. She notes that there is room for different search engines to fit user needs. She advocates email sorting based on multiple criteria.

Fagan [8] compares various studies for faceted browsing, their results, design and methodological practices.

Macdonald et al. [10] examine the retrieval performance of several different email fields present in the TREC15 email dataset, and find that the email body and subject are effective search fields, both individually and as a pair.

Dudek et al. [6] conduct a user study for web search engine usability. They find that different search egines fit different search needs, and that search speed is an important factor for participants.

Dos Santos Pergentino et al. [15] evaluate a search engine for the Brazilian Federal Court of Accounts, using a set of ten usability heuristics, and identify various usability issues.

Barnum et al. [3] compare searching in two versions of the same ebook. One version with an index with hyperlink references to pages. Another version without an index but with full-text search.

Harvey et al. [9] investigate behavioural patterns in an email search study in Thunderbird. They find that previous results are not good candidates for repeat queries in email search.

3 Theoretical analysis

This chapter examines the complexity of the program and establishes upper bounds for the running time, memory usage and disk usage of the mbox parser. The running time and memory of CompleteSearch's indexing data structure is explained in detail in [4]. The mbox parser runs before the indexer. The indexer process the output of the parser. Thus, we can add the running time and disk usage of both programs together. The memory usage is separate.

3.1 Time complexity

There are two scenarios to consider. In the first scenario, the mbox parser parses an mbox file and outputs a TSV file. In the second scenario, the mbox parser parses an mbox file, sorts the messages by date, and outputs a TSV file. In this section we analyze the time complexity in these two scenarios.

3.1.1 Unsorted output

Let n be the size of the input file. The parser needs to read the input file from disk. The time complexity of reading is trivially linear.

To write the output the parser must first transform each message into a TSV row. A message can be parsed in a single pass, but parts of the message might be encoded. The two encodings that are used in email message are base64 encoding and quoted-printable encoding.

Base64 encoding This encoding maps groups of six bits to 64 printable ASCII characters. Each ASCII character is a byte in size. The decoder reads a sequence of ASCII characters and decodes each character back into the group of six bits that it represents. Base64 decoding needs a single pass over the encoded input string.

Quoted-printable encoding Three printable ASCII characters are used to encode one byte. An arbitrary byte can be encoded as `=XX`, where each `X` can be `0-9` or `A-F`. Quoted-printable encoded data can be decoded in a single pass over the input string, replacing any `=XX` sequence to the corresponding byte value.

The Aho-Corasick algorithm is used to escape characters. Aho-Corasick needs $2k = O(k)$ state transitions to process an input of length k [1].

In this scenario, all components of the parsing process run in linear time with regards to the size of the input. Therefore, the upper bound on running time for unsorted output is $O(n)$.

3.1.2 Sorted output

Mbox files can be arbitrarily large. The parser cannot expect the entire mbox file to fit in working memory. Therefore, to be able to sort the messages of very large mbox files, we use an external sorter. An external sorter divides the data in segments of some fixed size δ. The sorter sorts each segment seperately in memory.

If the data do not fit in memory, sorted segments are written to disk. The messages in the input file are read into a buffer of limited size in memory. Once the buffer is full, the buffered messages get sorted and written to a sorted segment on disk. Afterwards, more messages can be read into the buffer and the process repeats. The entire input file is written to sorted segments on disk.

The sorted segments on disk are combined into a single sorted stream of messages when the segments are read back from disk. It is in essence a merge sort of the data by using the sorted segments on disk. The time complexity upper bound of merge sort is $O(x \log_2 x)$, where x is the number of items to sort. In this case, we are merge sorting email messages. Let d be the number of messages in the input mbox file. Then the upper bound is $O(d \log_2 d)$.

Thus, the upper bound on the running time of the parser is $O(n+d \log_2 d)$.

Let L be the average message length in the input file. The number of messages multiplied by the average message size is equal to the size of the input file. Thus, we can rewrite n as $d \cdot L$. Note that in practice, L can be considered a constant with respect to n. Average message length varies for different input files, but it does not grow or shrink in direct relation to n.

Therefore, the upper bound on time complexity for unsorted output is $d \cdot L = O(d)$. We say that $f(x) = O(g(x))$ as $x \to \infty$, if there exists a constant $C > 0$ and a starting point x_0, such that $f(x) \leq C \cdot g(x)$ for all $x \geq x_0$. In the case of sorted output, the time complexity is $L \cdot d + dlogd = O(d + d \log d)$. If we set $C = 2$, then $d + d \log_2 d \leq 2(d \log_2 d) = d \log_2 d + d \log_2 d$. This holds for all $x \geq 2$. Thus, $L \cdot d + logd = O(d \log d)$.

3.2 Memory usage

3.2.1 Unsorted output

To produce unsorted output, the parser only needs to keep a single message in memory at a time. The memory use is constant with respect to n.

3.2.2 Sorted output

For sorted output, the buffer of messages to be sorted must be kept in memory. Let k be the number of slots in the buffer. This value k is defined at compile time, so k is a constant. Let L_{max} be the maximum message length in the input file. Like L, L_{max} can be considered constant in relation to n. The memory usage is less than or equal to $k * L_{max}$, which are both constant. Therefore, memory usage for sorted output is also constant with respect to n.

3.3 Disk usage

3.3.1 Output TSV file

The parser transforms the information to another format. In case of unsorted output, messages do not depend on each other. The parser can construct a TSV record for each message in memory, one after the other. The parser can then write the record to the TSV output file before starting on the next message. The parser does not need to write any auxiliary data to disk. Therefore, the disk usage is linear in the size of the input file.

3.3.2 Sorting segments

To produce sorted output, all segments are written to disk. The segments contain all messages. Therefore, sum of all segments sizes is linear in the size of the input file. The segments are removed after sorting.

4 Implementation

At the highest level there is the Docker container and the Thunderbird program. The mbox parser and CompleteSearch run inside the Docker Container. The Prisma add-on is installed in Thunderbird. In the following subsection we will provide details for each component.

4.1 Mbox parser

The mbox parser is a program written in the Rust programming language[1]. It can efficiently parse an mbox file and output a format better suited for feeding into a search engine. Currently, the mbox parser can write output in TSV format or Turtle format. The parser filters the contents to the parts that are useful for text search. It selects specific header fields for each message. The parser converts text and html bodies, both plain ascii and transfer-encoded content, into utf-8 output text. The parser makes use of the mail-parser crate[2].

4.1.1 Dealing with attachments

Email attachment often contain binary data and cannot be directly searched as text. Additionally, attachments can be very large. Therefore, the parser strips attachments from the message by default. In most cases, stipping the attachments from messages shrinks the size of the resulting TSV output file considerably. Attachments names are preserved in the TSV output, so attachments can be searched by file name. CompleteSearch can parse the smaller resulting TSV file considerably faster. Stripping attachments reduces the time required for indexing and reduces the size of the TSV file and the CompleteSearch index.

[1]`https://www.rust-lang.org/`

[2]`https://docs.rs/mail-parser/latest/mail_parser/`

4.1.2 Maildir to mbox conversion script

Maildir archives can be converted into an mbox file. A simple way to do this, that is adequate for creating a searchable mbox archive, is simply to concatenate all files in the maildir, one after another, and inserting "From \n" separator lines in between. We supply a simple Nim[3] script that can be used to convert maildir to mbox (see appendix B).

4.2 Search engine

Prisma makes use of the CompleteSearch engine CompleteSearch is an efficient search engine for prefix and faceted search. CompleteSearch reads its input from a TSV (Tab Separated Values) file. In the case of Prisma, each record represents an email message. CompleteSearch parses the TSV file and creates a search index. The order in which messages occur in the TSV file determines the default ordering for the search results that CompleteSearch returns when it receives a query.

4.3 Container

The parser and the search engine run inside a Docker container. The container isolates these components from the host system. Input files must be explicitly mounted in the container to be processed. Any data that the components generate is stored in virtual volume, managed by Docker or Podman. This guarantees that Prisma cannot accidently delete or overwrite any file on the host system.

4.3.1 Run script

For development and debugging purposes, a `run.py` script is included in the implementation. The run script can be used to interact with the Docker container from the command line, without needing the add-on.

The basic invocation is as follows:

```
python run.py -f messages.mbox [action]
```

The available actions are: parse, index, start, parse-index, volume and *no action*, which is equivalent to parse-index-start.

[3]https://nim-lang.org/

Importantly, the script can also be used to build the container:

```
python run.py build
```

It allows Prisma to be combined with any tool or interface that can communicate via a local network port.

4.4 Setup and installation

We made an effort to simplify the setup steps for Prisma. The first important simplification is hiding container commands behind script (see section 4.3.1) in order to simplify the syntax that the user must remember. A design goal of Prisma is to work on both Linux and Windows. Therefore, another valuable simplification is making sure that the commands are identical between both operating systems.

We need a scripting language that works on both Linux and Windows. We want to avoid any dependencies that require an installer to be manually downloaded from a website. We prefer that the user stay within the control and safety of the operating system's own tools. On Linux, most users install software via a package manager from curated package repositories. However, downloading installers from the internet is a common practice on Windows. Luckily, Windows 11 comes with its own package manager: WinGet[4]. So if the scripting language is available on WinGet on Windows that is ideal. Importantly, we want to use a well-known scripting language. Although the user should ideally never have to look at any script, using a well-known scripting language assures that the user can find a large collection of online documents and websites in case they encounter a problem.

Given the above requirements for the scripting language, we chose to implement the script in the Python programming language[5]. First, Python is easy to obtain via system tools on both Linux and Windows.

Linux Python comes pre-installed by default on most Linux distributions. And if it is not on a specific distribution, it can easily be installed as a package. This makes it a good alternative to a Linux native shell script.

Windows 11 Python does not come bundled with Windows 11. However, it can be readily installed using the WinGet package manager. Alter-

[4]`https://github.com/microsoft/winget-cli`

[5]`https://www.python.org/`

natively, it can be installed via the Microsoft Store. Thus, Python is also easy and safe to obtain on Windows.

Additionally, Python has a large standard library and a modern, easy to read syntax. One downside of Python is that is does not integrate directly with the host operating system. A Docker container command cannot be invoked directly. The command needs to be invoked via the `subprocess` module, which is more verbose and less readable.

Compare the shell command:

```
docker build -t mycontainer
```

with the python equivalent using subprocess.check:

```
import subprocess
subprocess.check("docker build -t mycontainer")
```

Alternatively, there is the Docker SDK for Python[6]. However, the Docker SDK is not part of Python's standard library. Furthermore, the subprocess module is sufficient for Prisma's functionality. In order to keep the number of dependencies low, we decided to use the subprocess module.

4.5 Measurements

We measure the memory usage of the parser and indexer on Linux. We also measure the run time of the parser and indexer on Linux and Windows 11.

4.5.1 System information

We collect the measurements on the following systems:

Linux `Linux 6.10.1 SMP PREEMPT_DYNAMIC GNU/Linux`
`Lenovo YOGA 730-13IWL`
`Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz`
`2x8GB Row Of Chips DDR4 2400 MT/s`
`512Gb NVMe SAMSUNG MZVLB512HAJQ-000L2`

Windows `Windows 11 Home 23H2`
`HP Laptop 14s-dq2xxx`
`11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz`

[6]https://pypi.org/project/docker/

Table 4.1: Memory usage of the parser and the indexer

Component	Input	Usage
Parser	help-gnu-emacs	53M
Parser	qemu-devel	135M
Indexer	help-gnu-emacs	820M
Indexer	qemu-devel-500k	3.3G

```
1x8GB SODIMM DDR4 @ 3200 MT/s
256Gb NVMe KBG40ZNV256G KIOXIA
```

4.5.2 Mbox files

We use three different mbox files containing messages from three public GNU mailing lists[7]: poke-devel, help-gnu-emacs and qemu-devel. Each file contains all messages from its corresponding mailing list archive, until July 12th 2024 at 12:00 (PM). The poke-devel file has a size of 46M. The help-gnu-emacs file has a size of 683MB. The qemu-devel file has a size of 9.4GB. The indexer is tested on a subset of the qemu-devel file that has with the first 500,000 messages, with a size of 3.4G.

4.5.3 Memory usage

The data presented in this subsection were measured and collected using ProcPath[8]. Table 4.1 shows the memory usage of the parser and the indexer on Linux.

4.5.4 Run time

The data presented in this subsection were measured and collected using Hyperfine[9]. Table 4.2 shows the run time of the parser and the indexer on Linux and Windows 11.

[7]https://lists.gnu.org/

[8]https://heptapod.host/saajns/procpath

[9]https://github.com/sharkdp/hyperfine

Table 4.2: Run time of the parser and the indexer

Component	OS	File	Runs	Avg Time	σ
Parser	Linux	poke-devel	15	601.3 ms	47.1 ms
Parser	Linux	help-gnu-emacs	9	8.178 s	0.549 s
Parser	Linux	qemu-devel	3	150.103 s	1.044 s
Parser	Win11	poke-devel	15	3.852 s	0.262 s
Parser	Win11	help-gnu-emacs	9	37.769 s	3.640 s
Parser	Win11	qemu-devel	3	155.195 s	1.227 s
Parser	Win11 Docker	qemu-devel	1	602.211 s	
Indexer	Linux	poke-devel	5	7.549 s	0.319 s
Indexer	Linux	help-gnu-emacs	3	91.401 s	2.593 s
Indexer	Linux	qemu-devel-500k	1	762.306 s	
Indexer	Win11	poke-devel	5	8.194 s	0.083 s
Indexer	Win11	help-gnu-emacs	3	106.109 s	0.171 s
Indexer	Win11	qemu-devel-500k	1	903.331 s	

5 Methods

In this chapter we describe our evaluation approach. We conducted a four-person case study to asked users familiar with Thunderbird to compare Thunderbird's Global Search and Quick Filtering to the Thunderbird add-on of Prisma.

5.1 User study

The user study was conducted of the course of one week. The participants where selected with a combination of handpicked sampling and snowball sampling as explained by [13]. The participants were selected to fit three criteria:

- The participant has basic familiarity with Thunderbird.
- The participant uses email in a professional or work setting.
- The participant uses email on a daily basis.

Basic familiarity with Thunderbird was a hard requirement, else the participant could not compare Prisma to something they already know. Usage on a daily basis make sure that people have preferred ways of searching that can help them compare to alternatives. The idea about a profession setting is that people spend more time reading and sending email at work than at home. It meant that participants would compare Thunderbird and Prisma in a realistic scenario.

5.1.1 Data collection

The data was collected in the form of two questionnaires. The participants received an email with a brief overview of the user study (see appendix A.1 along with 4 PDF files with instructions A.2). Each participant could start at a day of their chosing, but were requested to finish both questionnaires within two weeks.

On the first day of the week, the participant was asked to install the necessary software on their computer and set up Prisma and the Thunderbird add-on. The participant was asked to describe this process in the introductory questionnaire.

During the week the participant was asked to use email as normal. When searching for an email the participant could use both Prisma and Thunderbird's integrated search. The participant could take notes of their experiences and form their opinions of the course of the week. On the last they of the week the particpant was asked to answer the main questionnaire.

5.1.2 Research questions

We are interested in the following five research questions:

RQ1 How does the usability of Prisma compare to Thunderbird Search?

RQ2 Do users prefer the search results of Prisma or Thunderbird Search?

RQ3 Which search features are preferred by users?

RQ4 Are Prisma or Thunderbird Search preferred in certain search scenarios?

RQ5 How is the first experience of Prisma for a user.

6 Results

In the first section of this chapter, we look at the responses given to the introductory and main questionnaires of the user study.

The questions and responses and additional raw materials of the user study are listed in A.3.

6.1 Introductory questionnaire

Three of the respondents think that the installation is quite easy. One respondent thinks that the installation process is complicated. Two repondents mention that they had to deviate slightly from the installation instructions. Three respondents say that the installation process was rather short and easy. One respondent seems to disagree. Another respondent thinks that the installation process is complicated. All respondents think that the supplied instructions were clearly written. Three repondents encounter minor problems during the installation process.

6.2 Main questionnaire

In this section we refer to Thunderbird Search as TBS and to the Prisma add-on as MSA.

Three respondents rate ease of use of TBS slighty higher than the ease of use of Prisma. One respondent prefers Prisma. One reason is that Prisma searches in one inbox at a time, whereas Thunderbird Global Search searches in all inboxes.

One respondent mentions that Thunderbird Global Search has clearer visual boundaries between search results. The presentation of TBS is better integrated with the rest of Thunderbird. One respondent thinks that the interface of MSA shows too many options.

The search results of Prisma were a bit better. R3 mentions that some results did show up in Prisma's results, but not in Thunderbird Search.

Accuracy of the results of Prisma and TBS is similar. Respondents are able to find results quicker with TBS. TBS integrates better with the rest of Thunderbird. The respondents mention that, as an add-on, MSA is less convenient to access. One respondent mentions that MSA's complex query features are more time consuming to use.

One repondent prefers MSA because it allows for quicker searching. Another respondent thinks that TBS is better for quick searching, whereas MSA is better suited for finding one specific email.

Three respondents prefer MSA or TBS for certain types of queries. Two respondents think that MSA's search functionality is more advanced, but both respondents mention that they have no use for the advanced functionality. All respondents think latency was good for both systems.

The respondents like the visual timeline of Thunderbird Global Search for its clarity and ease of use. Respondents also like the phrase queries. The dot is easier to type, but using quotes feels more familiar. One respondent likes proximity queries and lexicographic range queries. The other repondents do not use proximity queries or lexicographic range queries.

Throughout the week, respondents re-index their inbox either once a week, twice a week or every other day. Most respondent use the Unread filter of TBS Quick Filters. One respondent uses the Starred filter. Another respondent uses the Content filter. One respondent limits the number of indexed email messages. The other respondents indexes all emails in their inbox. The respondents think that the default settings of TBS and MSA are fine. The respondents do not feel a need to change the default settings. One respondent wants to see both the name and the email address in MSA, because it is safer with regards to spoofing.

One respondent prefers 'search as you type' in MSA because its quicker. Another respondent mentions that pressing Enter in Thunderbird Global Search gives them the feeling of being more in control. The respondents do not have strong preferences regarding automatic prefix search of MSA and the manual wildcard approach of TBS. Respondents think that the search features of MSA and TBS are sufficient for their purposes. One respondent says that TBS is easier to understand and use, but that MSA has useful query options.

7 Discussion

In this chapter we discuss the results of the user study and questionnaires with regards to the five research questions (see section 5.1.2).

7.1 User study

Regarding RQ1, the results seem to indicate that the respondents prefer the use of TBS to the use of Prisma. One reason is better integration of the search functionality with the rest of Thunderbird. It is important to note that MSA is at a disadvantage compared to Thunderbird's search functionality.

Prisma is an add-on, not a part of the standard Thunderbird interface. This means that Prisma is harder to find and reach. For example, the Thunderbird Global Search input field is always visible at the top of the Thunderbird interface. A user can click it from any tab or window and start searching. The Prisma add-on opens in a separate tab. A user has to switch to this tab in order to search with Prisma. So, convenience is likely part of the reason that the respondents preferred to search in TBS.

For the search results (RQ2), the distinction was less clear. Respondents seemed content with the search results of both systems. One important feature of Thunderbird Global Search was that it could search in multiple inboxes at once. Prisma only searches in one inbox at a time. The more advanced search features of Prisma did not convince the respondents.

The respondents indicate that Prisma's advanced search features made it more complex to use. Lastly, the visual timeline was a feature that was liked by all respondent. Prisma doesn't have a visual timeline. The phrase queries were also liked by the respondents. These features are both present in Thunderbird. For RQ3 it seems that repondents prefer the features that they are already familiar with.

In case of RQ4, the results indicate that Prisma was better suited for complex queries. TBS was easier to use and more familiar. Respondents do not mention needing Prisma's more advanced features to find their emails. This might be caused by the nature of email search, were the person who is searching has more information about what they are looking for, and queries are shorter. In this scenario, more advanced search features and prefix auto-completion might not provide a lot more value than a simple query. None of the respondent used the Cc, Bcc or Attachment preview field. The From, To and Date were the default, and most respondents did not change them. It might indicate that these fields do not provide enough information to justify the space they take up in the interface.

The first experience of Prisma (RQ5) was generally good, as indicated by the positive responses to the introductory questionnaire. However, one week did not convince the respondents to adapt MSA search features.

Multiple respondents mention that the features of MSA are too complex for their use case. The MSA interface is designed for more advanced search. Thunderbird's search interface is simpler and more familiar to the repondents. It is hard to convince users to change their habits.

8 Future work

In this chapter we mention several directions for improving Prisma. One direction is improving the user experience and search interface. Second, better integration with Thunderbird is important. A third direction would be implementing incremental indexing. Finally, searching in multiple inboxes at the same time would be a useful improvement.

8.1 Simplifying the search interface

The respondents of the user study mention their preference for simple features and an easy to understand search interface. Complex and advanced features work well for advanced users, but average users seem to prefer simplicity. The MSA interface presents a densely populated interface to the user. Spacing out the components could make the interface more appealing.

8.2 Visual features

The search interface of Prisma does not have many visual features. It is predominantly text-based. A visual timeline similar to Thunderbird Global Search would be a good addition. In combination with prefix autocompletion, the timeline could update dynamically as the user is typing their query.

8.3 Incremental indexing

Prisma's does not support incremental indexing. The reason for this is that the CompleteSearch engine does not implement incremental indexing. Incremental indexing would allow Prisma to append incoming messages to an existing index dynamically, rather than recomputing the entire search index to add one new message. This might require the search engine to be replaced.

For example, Tantivy[1] is an example of a search engine that implements incremental indexing. It is also written in the Rust programming language. Tantivy could be combined with the mbox parser into a single executable. A single executable would simplify Prisma's design. Additionally, the mbox parser could pass the message directly to the indexer, foregoing the need for intermediate TSV output.

8.4 Searching across multiple mbox files

The respondents mention that a downside of Prisma is, that it cannot search in multiple inboxes at the same time, whereas Thunderbird Global Seach can. It would be an improvement to Prisma's ease of use if Prisma supported searching in multiple mbox files at the same time. One way to implement this would be to allow mounting a directory in the Docker container. The parser could parse all mbox files in the mounted directory. The output for each mbox file could be appended to the same TSV file. One big TSV file could then be indexed by CompleteSearch. This would allow searching in all multiple mbox files at the same time with minimal changes to the Prisma's components.

[1]`https://https://github.com/quickwit-oss/tantivy`

References

[1] Alfred Aho and Margaret Corasick. “Efficient string matching: An aid to bibliographic search”. In: *Commun. ACM* 18 (June 1975), pp. 333–340. DOI: `10.1145/360825.360855`.

[2] Qingyao Ai et al. “Characterizing email search using large-scale behavioral logs and surveys”. In: *Proceedings of the 26th International Conference on World Wide Web*. 2017, pp. 1511–1520.

[3] Carol Barnum et al. “Index versus full-text search: a usability study of user preference and performance”. In: *Technical Communication* 51.2 (2004), pp. 185–206.

[4] Holger Bast, Christian W Mortensen, and Ingmar Weber. “Output-sensitive autocompletion search”. In: *Information Retrieval* 11 (2008), pp. 269–286.

[5] Holger Bast and Ingmar Weber. “The CompleteSearch engine: Interactive, efficient, and towards IR & DB integration”. In: *Third Biennial Conference on Innovative Data Systems*. 2007, pp. 88–95.

[6] Debra Dudek, Anna Mastora, and Monica Landoni. “Is Google the answer? A study into usability of search engines”. In: *Library Review* 56.3 (2007), pp. 224–233.

[7] European Parliament and Council of the European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council.* of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). May 4, 2016. URL: `https://data.europa.eu/eli/reg/2016/679/oj` (visited on 08/01/2024).

[8] Jody Condit Fagan. “Usability studies of faceted browsing: A literature review”. In: *Information Technology and Libraries* 29.2 (2010), pp. 58–66.

[9] Morgan Harvey and David Elsweiler. “Exploring query patterns in email search”. In: *European Conference on Information Retrieval.* Springer. 2012, pp. 25–36.

[10] Craig Macdonald and Iadh Ounis. “Combining fields in known-item email search”. In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval.* 2006, pp. 675–676.

[11] Joel Mackenzie et al. “Exploring user behavior in email re-finding tasks”. In: *The World Wide Web Conference.* 2019, pp. 1245–1255.

[12] Sebastian Michel and Ingmar Weber. “Rethinking email message and people search”. In: *Proceedings of the 18th international conference on World wide web.* 2009, pp. 1107–1108.

[13] Zina O’leary. *The essential guide to doing research.* Sage, 2004.

[14] Fabio Rutter. “Email Search Tool based on Associative Memory”. PhD thesis. Graz University of Technology, 2020.

[15] Ana Carolina dos Santos Pergentino et al. “Usability heuristics evaluation in search engine”. In: *Design, User Experience, and Usability. Interaction Design: 9th International Conference, DUXU 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part I 22.* Springer. 2020, pp. 351–369.

[16] Agnieszka Matysiak Szóstek. “‘Dealing with My Emails’: Latent user needs in email management.” In: *Computers in Human Behavior* 27.2 (2011), pp. 723–729.

[17] Steve Whittaker et al. “Am I wasting my time organizing email? A study of email refinding”. In: *Proceedings of the SIGCHI conference on human factors in computing systems.* 2011, pp. 3449–3458.

Appendices

A User study materials

A.1 Email to participants

Dear participant,

Thank you for taking part in this case study comparing two email search methods using the Thunderbird email program.

Study

The study takes place over the course of one week.

You are asked to compare two email search methods in Thunderbird:

1. Thunderbird's own Global Search and Quick Filters
2. Mailsearch, a search program and add-on developed for this study.

You should read and send emails as normal and compare the search methods over the course of the week.

Setup

The Mailsearch program that you need to install for this study can be downloaded from `link`.

Extract 'mailsearch-rev-25695.zip' to a location in your user home folder that you can easily remember. The ZIP archive contains the folder 'mailsearch-rev25695' which contains the Mailsearch source code. This folder is referred to as the Mailsearch project directory.

There are a few setup steps, which are explained in the attached PDFs:

1. Mailsearch requires some other programs to be installed. Information on how to install these required programs can be found in the attached file '1.

Installing Required Programs.pdf'.

2. Information on how to install Mailsearch itself can be found in the attached file '2. Installing Mailsearch.pdf'.

3. Information on how to use Mailsearch can be found in the attached file '3. Using Mailsearch.pdf'. This document goes over how to index your INBOX file so that you can search in it with Mailsearch.

4. A brief description of Thunderbird's Global Search and Quick Filters can be found in the attached file '4. Thunderbird Search.pdf'.

Questionnaires

As a participant, you are asked to fill out two questionnaires:

1. Introductory Questionnaire: Please answer these questions on the first day, when you install Mailsearch in Thunderbird. This questionnaire can be found online at `link`. Don't forget to submit your answers.

2. Main Questionnaire: Please answer these questions on the last day. Feel free to take notes during the week. This questionnaire can be found online at `link`. Don't forget to submit your answers.

If you log into a Google account, your questionnaire progress will be saved to your account even if it hasn't been submitted yet. If you don't have a google account then closing the questionnaire before you submit it will destroys all your progress, so please be careful.

I would very much appreciate it if you could fill out the introductory questionnaire by Sunday (April 7th), and fill out the main questionaire one week later, so on or before the following Sunday (April 14th).

If you have any questions regarding the study, the software or the questionnaire, please contact me via `<email-address>`.

Best regards,
Erik Schill
Student ID: es353
Uni Freiburg, Germany

A.2 Instructions

Installing required programs

Mailsearch works with Thunderbird 115 and later. Please make sure you update Thunderbird to the latest version.

In order to run mailsearch with Thunderbird, you need to install two other programs:

1. Docker, a container manager.
2. Python, a programming language (installed by default on Ubuntu)

This document contains installation instruction for Ubuntu and Windows 11.

Ubuntu

Install Docker

Docker is a program that manages containers on your system. See https://docs.docker.com/get-started for more information. You can install Docker with the following command:

```
sudo snap install docker
```

Running Docker as normal user

By default, Docker is only accessible with root privileges (sudo). To use docker as a regular user, you need to add your user to the docker group.

```
sudo addgroup --system docker
sudo adduser $USER docker
newgrp docker
sudo snap disable docker
sudo snap enable docker
```

Start the Docker daemon service manually

On Ubuntu, the Docker daemon service starts automatically. Use the following command to start it manually:

```
sudo systemctl start docker
```

On Ubuntu, the Docker service starts on boot by default. If it doesn't for you, run the following commands:

```
sudo systemctl enable docker.service
sudo systemctl enable containerd.service
```

See https://snapcraft.io/docker and https://docs.docker.com/engine/install for additional information.

Install Zenity

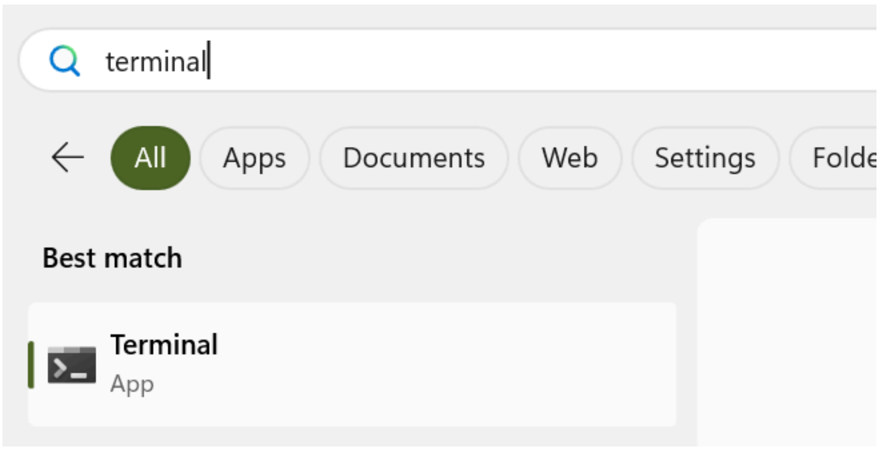
Zenity is used to implement a file selection dialog window. Ubuntu comes with `zenity` by default. You can install it manually with the following command:

```
sudo apt install zenity
```

Windows 11

The most reliable way to install the required programs is using the Terminal application.

Press ⊞ **Win** + **Q** to open the Start Menu. Enter ‘terminal’ in the search field and select the ‘Terminal’ application.



Install Python

In the terminal, you can install Python by typing the following command and pressing enter:

```
winget install -e --id Python.Python.3.12
```

Install Docker

Docker is a program that manages containers on your system. See https://docs.docker.com/get-started for more information. In the terminal, you can install Docker with the following command:

```
winget install -e --id Docker.DockerDesktop
```

If Windows Subsystem for Linux (WSL) isn’t installed, Docker Desktop will not be able to start the Docker Engine.

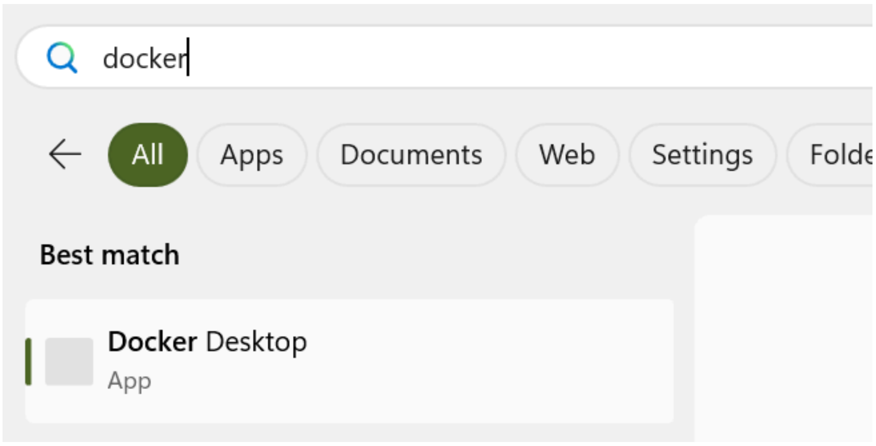
You can install WSL by running:

```
wsl --install
```

After the installation completes, restart you computer. When you sign in again, the installation should automatically continue.

Once WSL is installed you can open Docker Desktop.

Press ⊞ **Win** + **Q** to open the Start Menu. Enter ‘docker’ in the search field and select the Docker Desktop application.

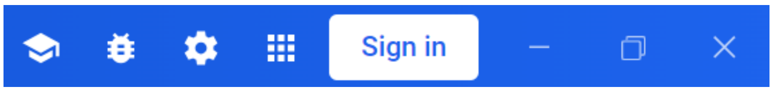

docker

Best match

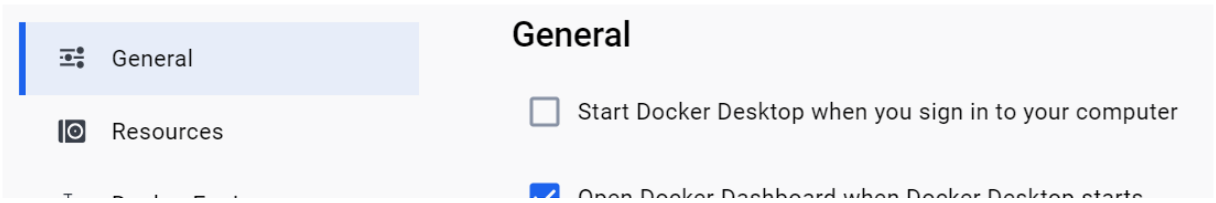
Docker Desktop
App

Skip creating an account, and the Docker Engine should start.

For convenience, you should set Docker Desktop to start on boot. Open the Settings by click the cogwheel icon in the top-right corner, left of the white Sign In button.



In General, enable Start Docker Desktop when you sign in to your computer.



General

☐ Start Docker Desktop when you sign in to your computer

☑ Open Docker Dashboard when Docker Desktop starts

If you don't enable this setting, you have to start Docker Desktop manually every time you sign out of your user account.

Installing Mailsearch

Make sure you have installed the required programs first. See '1. Installing Required Programs.pdf'.

We explain how to install Mailsearch on Ubuntu and Windows 11. There are two parts to the Mailsearch installation:

1. Installing the Mailsearch Server
2. Installing the Mailsearch Add-on in Thunderbird

Installing the Mailsearch Server

Ubuntu

Open a terminal and navigate to the Mailsearch project directory. The project directory contains the following files:

```
rk@arch ~/t/3/mailsearch-rev25695> ls
Dockerfile  README.md  applications/  entrypoint.sh*  extension/  install.bat  parser/  run.py*  setup.py*
```

Build the image

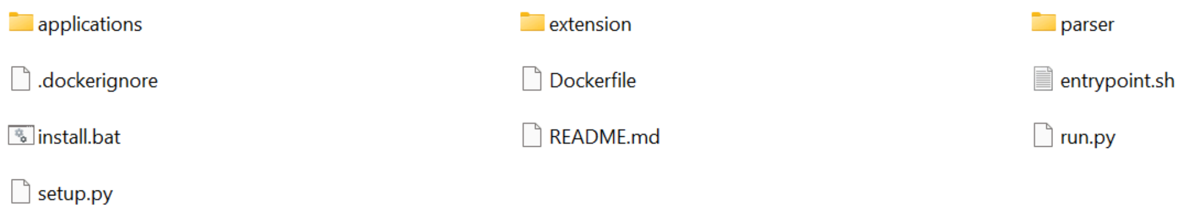
```
python run.py build
```

Set up native messaging

```
python setup.py
```

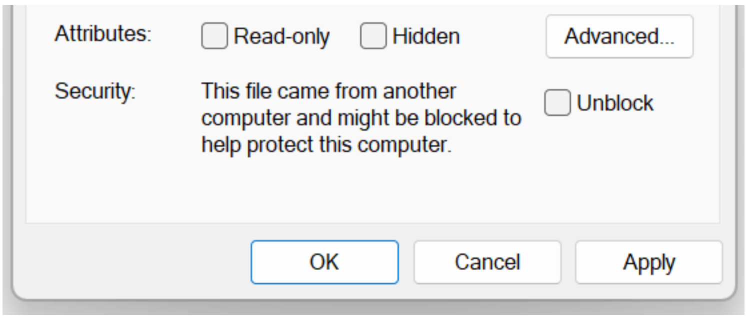
Windows 11

Navigate to the Mailsearch project directory.



```
applications        extension        parser
.dockerignore       Dockerfile       entrypoint.sh
install.bat         README.md        run.py
setup.py
```

Right-click on `install.bat` and select Properties. At the bottom of the Properties window, check Unblock and then click OK.



Attributes: Read-only Hidden

Security: This file came from another computer and might be blocked to help protect this computer. Unblock

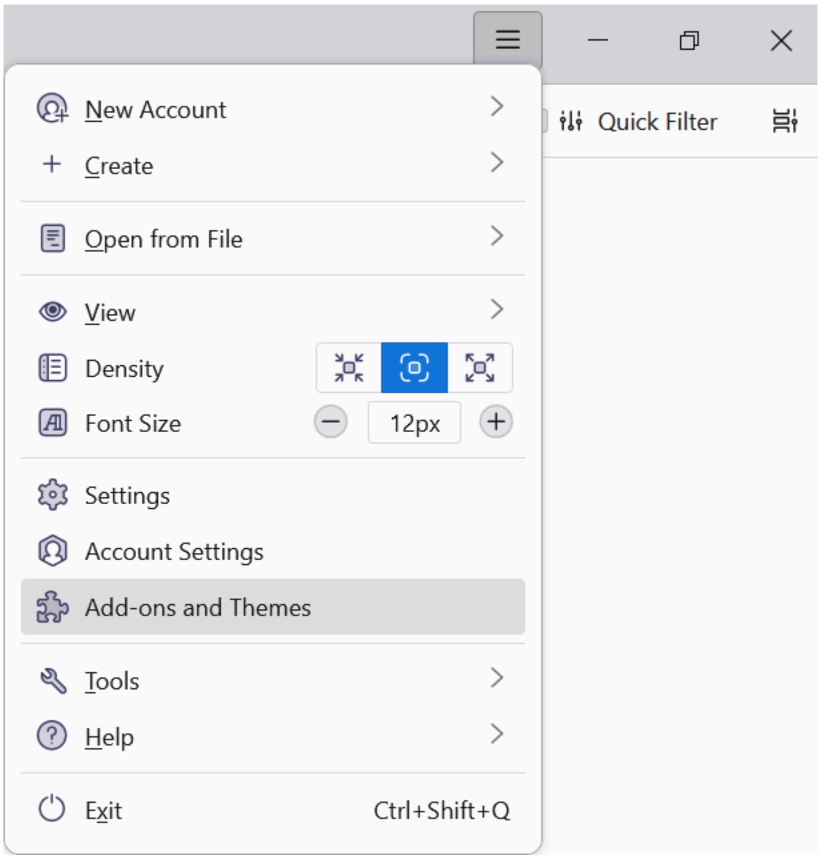
Build the image and set up native messaging

Double click on `install.bat` to execute the script. The script will build the image and set up native messaging. When the script completes you can press enter to close the terminal window.

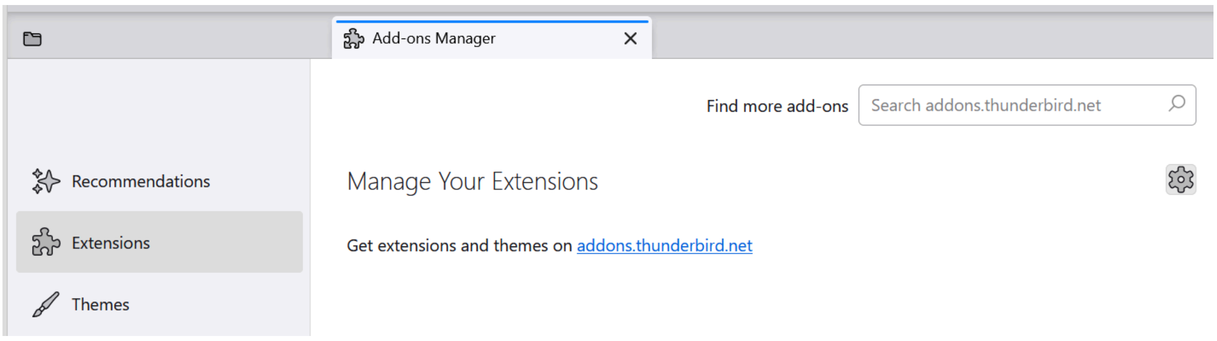
Installing the Mailsearch Add-on in Thunderbird

This part is identical on Ubuntu and Windows 11.

Open Thunderbird. Click the ☰ (hamburger) button in the top-right to open the menu, then select Add-ons and Themes.



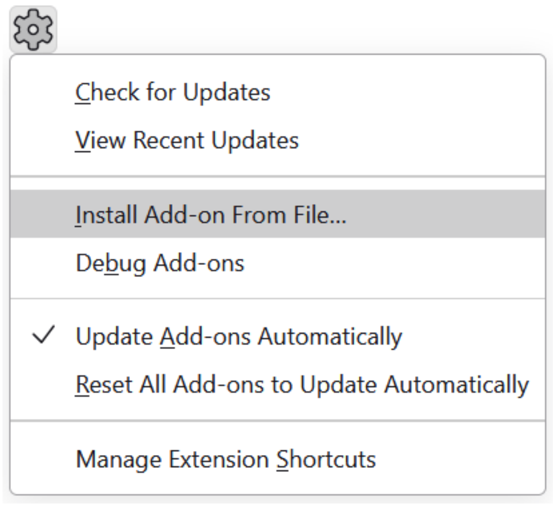
In Add-ons and Themes, click on Extensions in the side bar to open the Add-ons Manager.



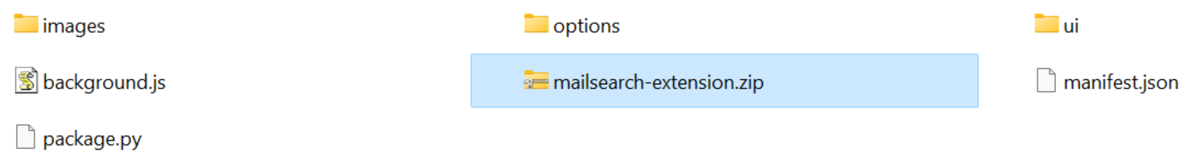
Manage Your Extensions

Get extensions and themes on addons.thunderbird.net

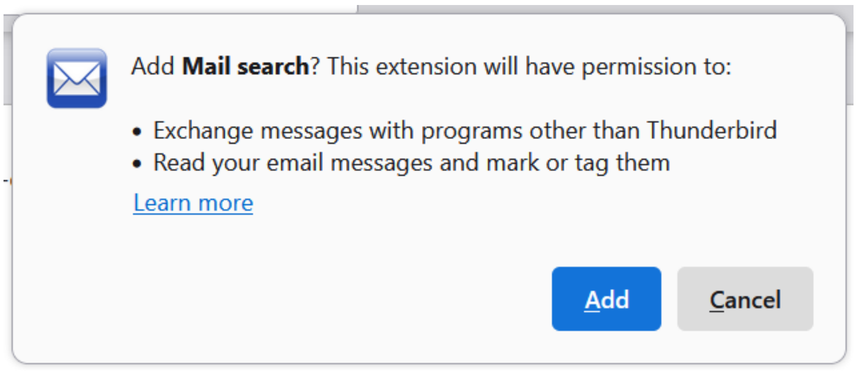
In the Add-ons Manager, click the cogwheel button in the top-right and select Install Add-on From File.


Check for Updates
View Recent Updates
Install Add-on From File...
Debug Add-ons
✓ Update Add-ons Automatically
Reset All Add-ons to Update Automatically
Manage Extension Shortcuts

A file selection dialog will open. Navigate to the Mailsearch project directory and go into the `extension` subdirectory. In the `extension` subdirectory, open `mailsearch-extension.zip`.



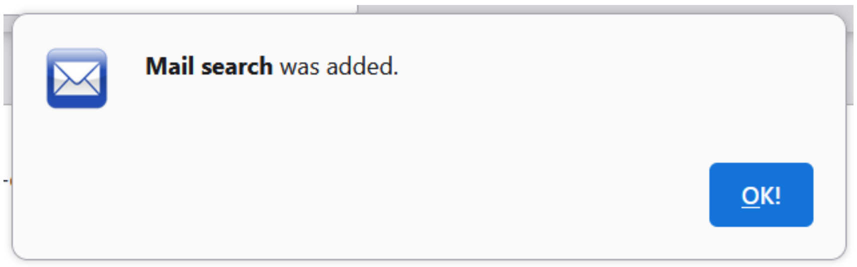
You will be prompted to grant the extension access to the required permissions.



Add **Mail search**? This extension will have permission to:

- Exchange messages with programs other than Thunderbird
- Read your email messages and mark or tag them

Click the Add button. Another popup will open to tell you the extension was installed.



Mail search was added.

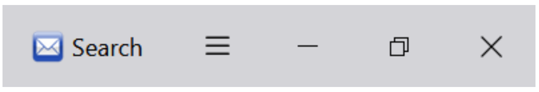
Click the OK button.

Using Mailsearch

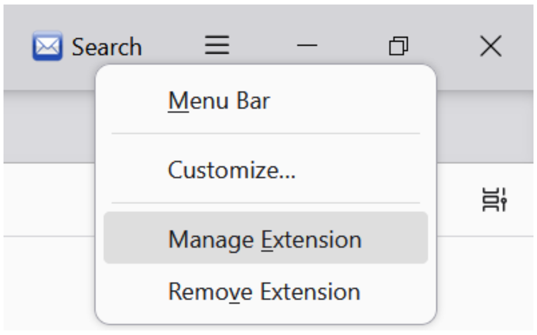
Mailsearch has an options page and an search page. In the options page you can index an MBOX file and change several settings. In the search page you can search in the MBOX file once it has been indexed and the server is running.

In order to use Mailsearch, you must first install it. See '2. Installing Mailsearch.pdf'.

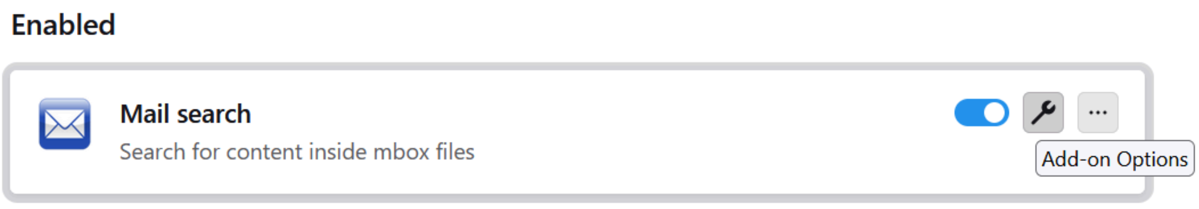
Once installed, there should be now be a Search button in the Mail View. You can press `Ctrl+1` (one) to go to the Mail View.



If you click the Search button you go to the search page. But we first need to index an MBOX file. So, right-click on the Search button and select Manage Extension.

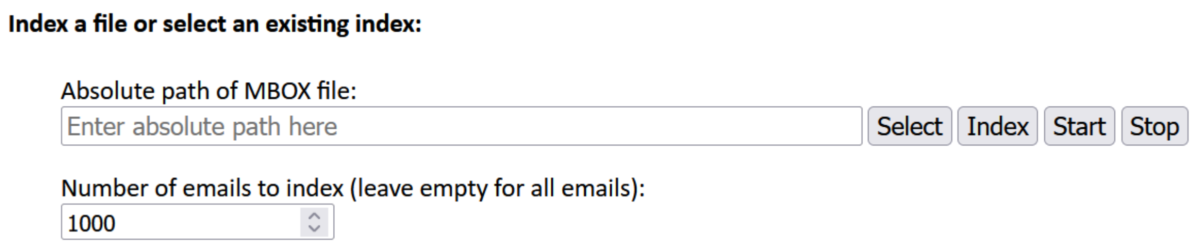


This opens the Add-On Manager entry for the extension. You can open the options page by click the wrench button.



Options page

In the options page you can index a file, and change a few options that affect the search page.

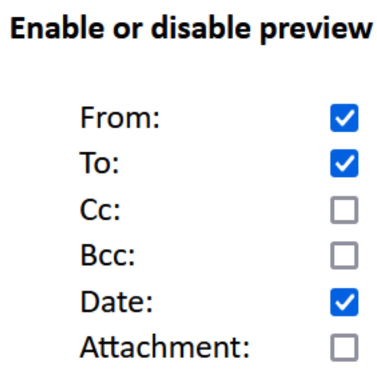

Index a file or select an existing index:
Absolute path of MBOX file:
Enter absolute path here
Select
Index
Start
Stop
Number of emails to index (leave empty for all emails):
1000

You can press the Select button which opens a file selection dialog where you can select the file you want to index. Alternatively, you can copy the absolute path to the file and paste it into the input field, that says 'Enter absolute path here'. Then click the Index button to start indexing. You can index multiple files (one at a time), but you can only search in one file at a time. If you want to switch to another index without re-indexing the file, clear the input field and click the field again. This reveals a list of previously indexed files. Click on the file that you want, then click the Start button. The Stop button allows you to stop the server or the active indexing process.

By default, only the 1000 most recent emails will be indexed. You can enter a different number into the input field below 'Enter the number of emails to index'. You can also leave the input field empty to index all emails, but for large INBOX files this is not advised.

The input field turns blue when indexing. It turns green when indexing was successfully. On error, the input field turns red.

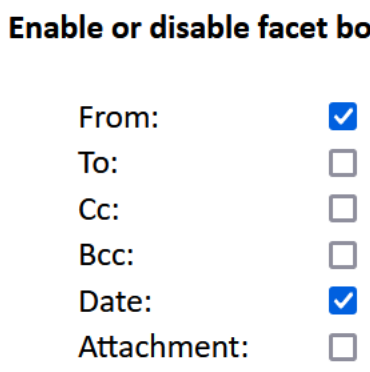
In case of an error, you can check the log messages. Press `Ctrl+Shift+i` to open the Developer Tools. You will be prompted to allow the incoming connection. Click 'OK' to allow the connection. The log messages are shown under the Console tab. **Warning:** Note that the Developer Tools slow down Thunderbird, so keep the the Developer Tools closed under normal use.



Enable or disable preview fields:

From:	☑
To:	☑
Cc:	☐
Bcc:	☐
Date:	☑
Attachment:	☐

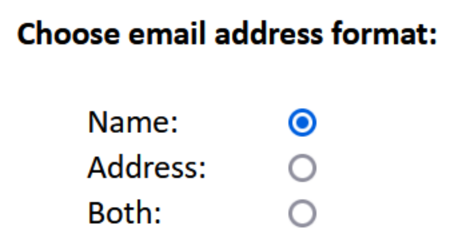
Here you can enable and disable preview fields. If you enable a field, it will show in every result that contains the field. If you disable the field, it will not be shown in any result. Enabling more fields will show more information, but a result will take up more visual space.



Enable or disable facet boxes:

From:	☑
To:	☐
Cc:	☐
Bcc:	☐
Date:	☑
Attachment:	☐

Here you can enable or disable facet boxes. If you enable a facet box, you can use its suggestions. If you disable a facet box, you cannot use its suggestions and it takes up no visual space.

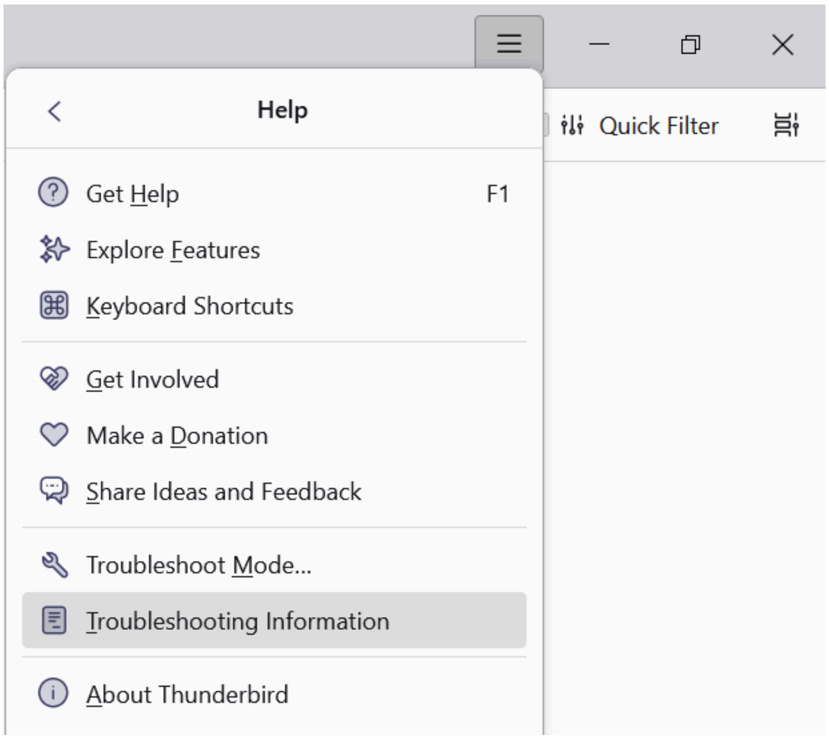

Choose email address format:

Name: [✓]
Address:
Both:

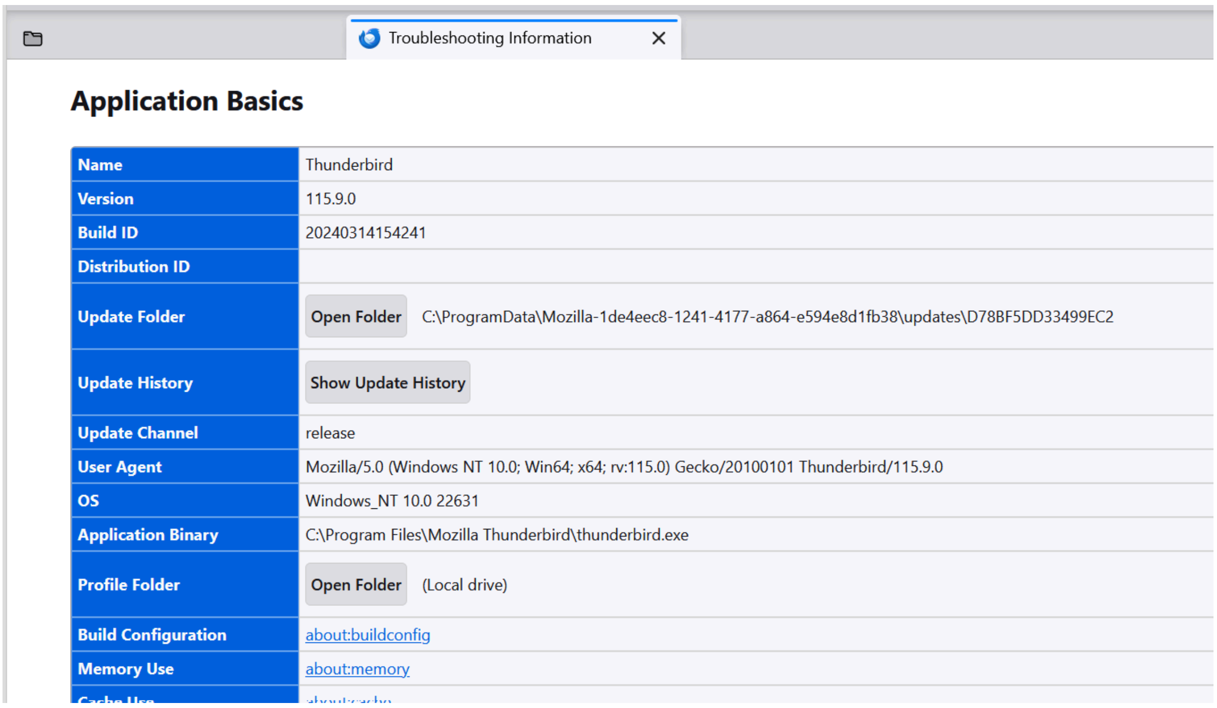
Here you can choose what email address format you prefer for the facet boxes. Selecting Name will display only a person’s name if it is available. Selecting Address will only display the email address. Selecting Both will display both the name and the address if both are available.

Finding your INBOX file

You will need to index your INBOX file so you can search in it with Mailsearch. The INBOX file is located in your profile directory. You can find the profile directory by clicking the `☰ > Help > Troubleshooting Information`.



In the troubleshooting page, find the Profile Folder entry and click the Open Folder button.


Troubleshooting Information

Application Basics

Name	Thunderbird
Version	115.9.0
Build ID	20240314154241
Distribution ID	
Update Folder	Open Folder C:\ProgramData\Mozilla-1de4eec8-1241-4177-a864-e594e8d1fb38\updates\D78BF5DD33499EC2
Update History	Show Update History
Update Channel	release
User Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:115.0) Gecko/20100101 Thunderbird/115.9.0
OS	Windows_NT 10.0 22631
Application Binary	C:\Program Files\Mozilla Thunderbird\thunderbird.exe
Profile Folder	Open Folder (Local drive)
Build Configuration	about:buildconfig
Memory Use	about:memory

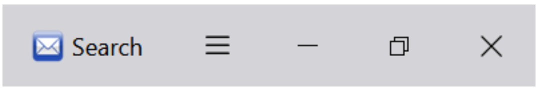
This will open a File Explorer window. Go to the ImapMail directory. In the ImapMail directory, go to the subdirectory that matches your email address. Inside this subdirectory is your INBOX file. Select the INBOX file, then copy its path with `Ctrl+Shift+c`. In the Mailsearch options page, paste the copied path into the absolute path input field. Then click the Index button to start indexing.

Re-indexing your INBOX file

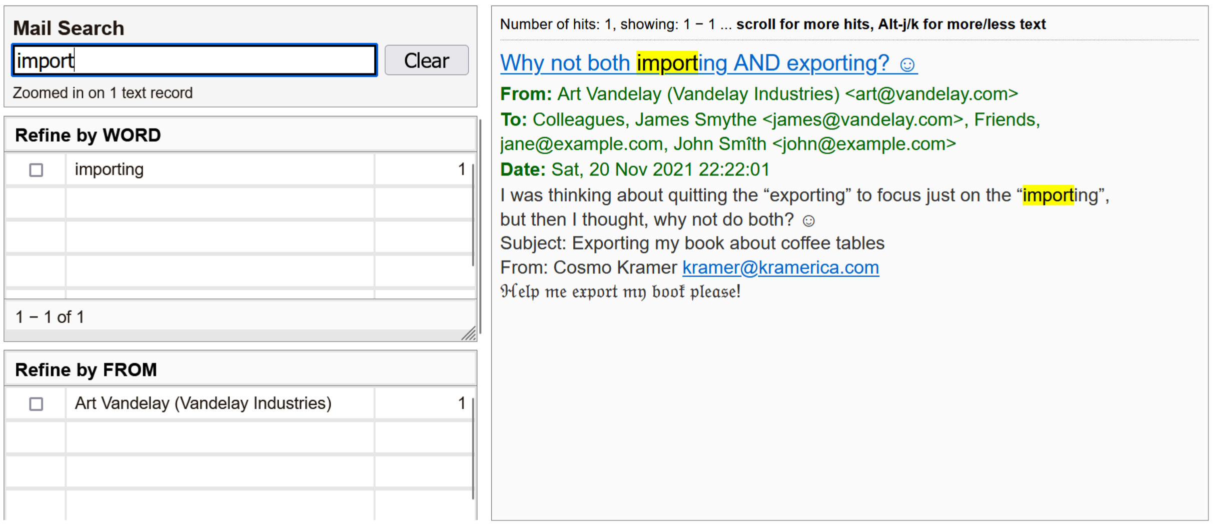
You can press the Index button in the options page at any time to re-index you INBOX file. Re-indexing will make the latest emails available as results in the search page.

Search page

You can go to the Mail View with `Ctrl+1` (one) and click the Search button in the top-right corner.



This opens the search page.


Mail Search

import

Clear

Zoomed in on 1 text record

Refine by WORD

☐	importing	1

1 – 1 of 1

Refine by FROM

☐	Art Vandelay (Vandelay Industries)	1

Number of hits: 1, showing: 1 – 1 ... **scroll for more hits, Alt-j/k for more/less text**

Why not both importing AND exporting? ☺

From: Art Vandelay (Vandelay Industries) <art@vandelay.com>
To: Colleagues, James Smythe <james@vandelay.com>, Friends, jane@example.com, John Smîth <john@example.com>
Date: Sat, 20 Nov 2021 22:22:01
I was thinking about quitting the “exporting” to focus just on the “importing”, but then I thought, why not do both? ☺
Subject: Exporting my book about coffee tables
From: Cosmo Kramer kramer@kramerica.com
Help me export my book please!

In the top-left corner, you see the input field with the Clear button. You can type your queries in the input field and results will show up automatically. Matched words will be highlighted. As you type, you can press `Alt+j` and `Alt+k` to show more or less text per result. The Clear button clears the query.

On the left, below the input field, you can see the facet boxes. The facet boxes are titled `Refine by FROM` and `Refine by DATE` to indicate what email field they correspond to. The facet boxes will provide dynamic suggestions as you type. The `Refine by WORD` facet box provides completions for the last word in the input field.

On the right are the results. Each result starts with a blue Subject line, which is also a link. You can click the link to open the result in a separate Thunderbird window, where you can easily reply to the message.

Types of queries

CompleteSearch performs prefix search by default. So 'import' will match both 'import' and 'importing' etc. There are several types of queries:

- **Word query**: The basic query that tries to match each word separately, and scores results based on the number of words that match.
- **Filter query**: It is possible to restrict the query for a word to a specific field. For example, to only match messages that have 'bank' in the subject, you can use 'subject:bank'. The available filters are 'subject:', 'from:', 'to:', 'cc:', 'bcc:', 'date:', 'attachment:'.
- **Phrase query**: You can seperate words by '.' (period) to indicate that the words must occur next to each other. For example, 'big.apple' would match 'big apple' but not 'big green apple'.
- **Proximity query**: You can also separate words by '..' (double period) to indicate that the words should occur close together, but not necessarily right next to each other.
- **Lexicographic range query**: You can write 'boaa--bozz' to match all prefixes between 'boaa' and 'bozz', such as 'boat', 'book', 'bold' etc.

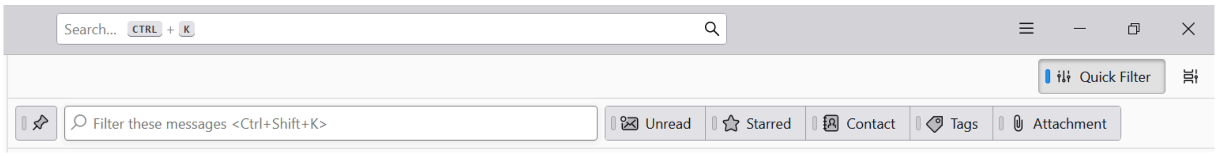
These queries can be used together in the search field. For example, 'big.apple city' will match the 'big apple' as a phrase, and will match 'city' separately.

Thunderbird Search

Here we briefly describe which components of Thunderbird's search functionality we want to compare Mailsearch with.

We are interested in the following two components:

1. Global Search
2. Quick Filter Bar



Global Search

Global Search is located at the top of the Thunderbird window (visible in picture above). You can access it with `Ctrl+k`.

See https://support.mozilla.org/en-US/kb/global-search for more information.

Quick Filter Bar

Quick Filters are used to search messages in a specific message list. You can toggle the Quick Filter Bar by clicking the Quick Filter button in the top-right corner (visible in picture above). The Quick Filter Bar will then be visible above your message list. You can press `Ctrl+Shift+k` to access it.

See https://support.mozilla.org/en-US/kb/quick-filter-toolbar for more information.

A.3 Responses

PDF versions of the raw data for readability
Raw data files: `intro_responses.csv`, `main_responses.csv`

Numerical answers to 'How X was ...' questions where selected by the respondents on an integer scale from 0 (left) to 10 (right), with the left labeled 'Very {negative X}' and the right labeled 'Very {positive X}'.

Note: the documents in this appendix refers to Mailsearch instead of Prisma. They refer to the same tool. Mailsearch was the working title as the time.

Both questionnaires began with this message:

Thank you for participating in this case study!

You are asked to compare two methods of searching emails in the Thunderbird email program:

1. Thunderbird's global search and quick filters, after this point abbreviated as **TBS**.
2. Mailsearch program and add-on for Thunderbird developed for this study, after this point abbreviated as **MSA**.

For the introductory questionnaire the message ended with:

This is the **Introductory Questionnaire**. Please answers these quesions on the **first day** of the week.

For the main questionnaire the message ended with:

This is the **Main Questionnaire**. Please answers these quesions on the **last day** of the week.

A.3.1 Introductory questionnaire responses

Q1. How old are you?

Respondent 1 32

Respondent 2 34

Respondent 3 25

Respondent 4 74

Q2. What is your profession?

Respondent 1 Sales Support

Respondent 2 GIS specialist

Respondent 3 PhD Mathematics

Respondent 4 business consultant

Q3. What operating system are you using?

Respondent 1 Windows 10

Respondent 2 Fedora Linux 39

Respondent 3 Windows 11

Respondent 4 Windows 11

Q4. What is the number of messages in your inbox?

Respondent 1 7444

Respondent 2 1051

Respondent 3 6409

Respondent 4 7296

Q5. What is the size on disk of your inbox?

Respondent 1 3.9 GB

Respondent 2 160MB

Respondent 3 1,8GB

Respondent 4 4,1 GB

Q6. What version of Thunderbird are you using?

Respondent 1 115.9.0 (64-bit)

Respondent 2 115.9.0

Respondent 3 115.9.0 (64-bits)

Respondent 4 115.9.0 64 bit

Q7. How difficult was the installation of the Mailsearch server and the Thunderbird add-on?

Respondent 1 9

Respondent 2 8

Respondent 3 8

Respondent 4 5

Q7.1. Please explain your choice

Respondent 1 All the steps were clearly and extensively explained in the attached document. It was easy to install.

Respondent 2 Most of the dependencies were already available on my system and I'm not unfamiliar with using the terminal to install software or to run scripts. I deviated from the instructions slightly, as I don´t use Ubuntu and do not have snapd installed on my system. As a long-time user of Mozilla software, installing add-ons is also a familiar process for me.

Respondent 3 It was a pretty short and easy installation aside from my own misreading's of a couple of steps. One improvement I can think of is showing which tab is open on the last picture of the first page.

Respondent 4 It was doable, but I doubt whether such a rather complicated installation could be a salable product.

Q8. How clear were the installation instructions for Mailsearch?

Respondent 1 7

Respondent 2 9

Respondent 3 7

Respondent 4 8

Q8.1. Please explain your choice

Respondent 1 All the steps and options were clearly and extensively explained in the attached document.

Respondent 2 All the steps were laid out and accompanied by screenshots to help guide me

Respondent 3 Some steps missed some pictures and some pictures could be improved with some highlights. But with the help of the text everything was clear.

Respondent 4 The instructions were very clear and complete. I'd never have managed the installation without them.

Q9. Did you encountered any errors during the installation process?

Respondent 1 Yes

Respondent 2 Yes

Respondent 3 No

Respondent 4 No

Q9.1. If yes, what was the error you encountered?

Respondent 1 I had difficulty to copy the path of my INBOX file correctly. Therefore, indexing my INBOX file needed some retries but it now works properly and turned green.

Respondent 2 Docker wasn't running the first time I tried installing the Mailsearch server. This was due to user error: I did enable the systemd Docker service to run at startup, but I accidentally skipped the step to start the service right away.

Respondent 3 Not really an error, but I got some weird security notification. Didn't have an impact on the installation procedure as far as I am aware however.

Respondent 4

A.3.2 Main questionnaire responses

Q1.1. TBS: How was the ease of use?

Respondent 1 8

Respondent 2 9

Respondent 3 7

Respondent 4 7

Q1.2. MSA: How was the ease of use?

Respondent 1 7

Respondent 2 7

Respondent 3 8

Respondent 4 6

Q1.3. Please explain your choices

Respondent 1 Both are easy to use and often give the desired results quickly, even in my rather large mailbox of more than 5000 emails. I use TBS more often because it's ready to use on the main Thunderbird tab, while for TBS I need to first open the add-on seperately. Secondly, MSA only searches in the indexed inbox, while TBS searches in all folders (with Global Search), or one specific folder (Quick Filter Search). Thirdly, there is the visual difference in how the two search methords show the results. In my opinion, with TBS it's easier to distinquish between different results because of the horizontal. What I liked about MSA, in comparison to TBS, is that I can do multiple searches after each other in one tab, while with TBS every search opens a new tab.

Respondent 2 TBS is more ingrained in Thunderbird, so using it feels a bit more natural. I quite like the distinct search options TBS provides (global search and filter), because that allows me to make a search query in the relevant section of my mailbox immediately. If I'm already browsing in a certain folder, it makes sense to have a quick search in that specific folder (but also having an option to search globally nearby). The search results in TBS are more neatly organized. It is easier to distinct separate emails whereas in MSA it is a little harder to see where one result ends and another starts.

I like the "refine by word" MSA provides, but the filter by date-option is too detailed to be of any use.

Respondent 3 Kind of annoying that you have to click twice to filter results for TBS. Aside from this both search engines were easy to use.

Respondent 4 There are more options in MSA than I'll ever need, making MSA more complicated. In TBS, the time line was a practical feature.

Q2.1. TBS: How good were the search results?

Respondent 1 8

Respondent 2 8

Respondent 3 6

Respondent 4 8

Q2.2. MSA: How good were the search results?

Respondent 1 8

Respondent 2 7

Respondent 3 8

Respondent 4 8

Q2.3. Please explain your choices

Respondent 1 With most searches, both methods give me the desired and the same results.

Respondent 2 Neither gave me too much trouble finding my emails when using regular queries. Phrase queries also provided the expected results in both search options. I've found the proximity and lexicographic range query options in MSA to give unpredictable results. This might be due to my unfamiliarity with these type of queries, but I got all kinds of results I didn't expect nor were the emails I was looking for.

Respondent 3 It seemed like TBS didn't always give all the relevant results, both global and quick search. MSA often gave the ones from TBS and more and the extra results were always for as far as I am aware relevant. Although there were times where TBS gave more relevant mails

Respondent 4 Both were good enough for my purposes

Q3.1. TBS: How was the accuracy of the search results?

Respondent 1 8

Respondent 2 8

Respondent 3 6

Respondent 4 8

Q3.2. MSA: How was the accuracy of the search results?

Respondent 1 8

Respondent 2 7

Respondent 3 8

Respondent 4 8

Q3.3. Please explain your choices

Respondent 1 Both methods give me similar results and the desired results with most of my searches. Accuracy of the results is mostly dependent on how specific my own input is.

Respondent 2 I haven't noticed a difference in accuracy.

Respondent 3 See my explanation of the previous question.

Respondent 4 I didn't see much difference in the accuracy

Q4.1. TBS: How fast were you able to find the emails you were looking for?

Respondent 1 8

Respondent 2 9

Respondent 3 7

Respondent 4 8

Q4.2. MSA: How fast were you able to find the emails you were looking for?

Respondent 1 7

Respondent 2 7

Respondent 3 8

Respondent 4 7

Q4.3. Please explain your choices

Respondent 1 TBS is a bit faster for me, first because it's integrated in the main Thunderbird mailbox tab. Second, because I can choose between Global Search or Quick Filter Search and thirdly because I like the Toggle Timeline feature. With both TBS and MSA I use the facets to find my desired results, but the 'Refine by DATE' facet in MSA is too cumbersome compared to the Toggle Timeline in TBS.

Respondent 2 TBS and its option to quickly switch between searching in your current folder and searching globally is a massive benefit in the speed in which to find your emails. The better layout of the results also helps. In MSA I found myself scrolling up and down the results to find the actual email I was looking for only to notice I missed it in first time passing.

Respondent 3 All my e-mails could be found relatively fast with both engines. I might have been lucky however that the discrepancies mentioned in the previous questions did not impact the e-mails I was trying to find. Searching numbers in TBS did not seem to work in global search.

Respondent 4 Setting and finetuning the queries was a bit more time consuming in MSA

Q5. Do you prefer TBS over MSA, or vice versa?

Respondent 1 I prefer TBS

Respondent 2 I prefer TBS

Respondent 3 I prefer MSA

Respondent 4 I do not prefer one over the other

Q5.1. Please explain your choice

Respondent 1 First, because TBS is better integrated in the main Thunderbord program. Secondly, I prefer the visual layout of showing results in TBS. Thirdly, because of Global Search and Quick Filter Search instead of only the indexed inbox. The different types of queries possible in MSA might be really useful for others, but with how I use my email and how I search for emails I don't seem to ever use.

Respondent 2 It suites (maybe due to familiarity and the way I use my email) my work flow a little better. The more advanced search queries MSA provides do not provide benefits for my type of email queries.

Respondent 3 MSA combined the pros of the quick search and the global search and I barely experienced any cons.

Respondent 4 It depends on what type of searching you need. TBS is easier for quick and dirty searching, MSA may be better for detailed research for that one email that you should absolutely find.

Q6. Was either TBS or MSA more useful for certain types of queries?

Respondent 1 No

Respondent 2 No

Respondent 3 Yes, MSA

Respondent 4 Yes, both TBS and MSA were more useful for certain types of queries

Q6.1. Please explain your choice

Respondent 1 MSA has more advanced types of queries but I don't use them at all, so therefore don't create an advantage over using TBS.

Respondent 2 As explained above, MSA does provide more advanced types of queries, but I did not find myself using those very much due to unpredictable results. Therefore these were not useful. In regular queries there wasn't too much distinction, so neither were more useful.

Respondent 3 Searching numbers in TBS did not seem to work in global search as explained previously.

Respondent 4 See my explanation under question 5.

Q7.1. TBS: How was the latency while searching?

Respondent 1 5

Respondent 2 9

Respondent 3 9

Respondent 4 8

Q7.2. MSA: How was the latency while searching?

Respondent 1 5

Respondent 2 10

Respondent 3 9

Respondent 4 8

Q7.3. Please explain your choices

Respondent 1 I don't notice any difference in latency. MSA shows live results as you type, while TBS does not, but this does not influence how I search my queries.

Respondent 2 Both pretty fast with no discernible delays and at least not of a length that breaks work flow.

Respondent 3 Could not detect much difference. Did not experience any latency issues with both.

Respondent 4 Latency good in both MSA and TBS according to my standards.

Q8.1. TBS Global Search: How helpful were the facet boxes?

Respondent 1 7

Respondent 2 9

Respondent 3 8

Respondent 4 8

Q8.2. MSA: How helpful were the facet boxes?

Respondent 1 8

Respondent 2 6

Respondent 3 8

Respondent 4 8

Q8.3. Please explain your choices

Respondent 1 I am an interface user and not so much a typer of extensive specific search queries so I definitely like the facet boxes. In TBS I do like the 'Toggle Timeline' facet while in MSA I like the 'Refine by WORD' and 'Refine by FROM' facet boxes. In TBS I don't seem to use the 'People' facet often, I prefer MSA to specify who the email is from. Compared to TBS's Toggle Timeline, I feel "Refine by DATE" is too specific for me and therefore I don't use it.

Respondent 2 TBS: helpful, though the addition of a word filter caret as in MSA would be nice. The timeline filter is very nice and intuitive to use and drill through your search results. MSA: both a 9 and a 2: the word filter caret is a nice way to overrule the prefix search (though it would also be nice to have a query to do so). The date filter however is way too detailed to be useful: I generally don't know the precise minute an email arrived.

Respondent 3 Both are very different, but I did not really have a preference.

Respondent 4 TBS facet boxes seem to be more helpful for global searching, MSA for detailed searching.

Q9.1. TBS Quick Filters: How helpful were the query filters?

Respondent 1 8

Respondent 2 8

Respondent 3 4

Respondent 4 7

Q9.2. MSA: How helpful were the query filters?

Respondent 1 6

Respondent 2 6

Respondent 3 7

Respondent 4 3

Q9.3. Please explain your choices

Respondent 1 Again, I am an interface user and not so much a typer of extensive specific search queries. Therefore I definetely prefer TBS Quick Filters over MSA. Especially after having started a search querie, I am focused on the shown results and using my mouse and don't feel like going back to type a new or amended querie.

Respondent 2 I found myself using them more in TBS than in MSA bacause I use them after my inital search query. I type my query, hit enter and then switch my hand to my mouse to scroll through the results. Then it is easier/quicker to click a query filter ("Subject") than to click the search box again and type "subject:" to my query.

Respondent 3 TBS quick search query filters seemed a bit clunky. MSA filters were quite easy to use. Only downside is that you sometimes need to look up how you use them. A list of options on screen somewhere would have been nice, for example a small help button which gives a pop-up with the options for search queries.

Respondent 4 For my purposes the MSA query filters are much too complicated in use. I prefer the quick and dirty solution of TBS.

Q10.1. TBS Global Search: How helpful were the phrase queries? (example: "big apple")

Respondent 1 8

Respondent 2 9

Respondent 3 7

Respondent 4 0

Q10.2. MSA: How helpful were the phrase queries? (example: big.apple)

Respondent 1 8

Respondent 2 8

Respondent 3 8

Respondent 4 7

Q10.3. Please explain your choices

Respondent 1 I don't notice a different specifically on this aspect. Both programs work with a period between the words (big.apple). In general phrase queries can be very useful to specify a querie.

Respondent 2 I did find myself sometimes mistyping the phrase queries in MSA. Due to 20 years of using " in search engines on the internet, using periods took a little while to get used to. I did not like how big and apple in big.apple show up in two colours in the search results. I feel like this should be one colour to represent one part of the query.

Respondent 3 Slightly easier to use a point than quotation marks, but not noteworthy. Downside for both is that you can't search for words with a point between up (for MSA), like in links and quotes (for TBA). At least as far as I am aware.

Respondent 4 I succeeded to get results with phrase queries in MSA, but not in TBS.

Q11. How helpful was the timeline of TBS Global Search?

Respondent 1 8

Respondent 2 9

Respondent 3 7

Respondent 4 9

Q11.1. Please explain your choice

Respondent 1 Very helpful indeed. Specifying a year you want to see results from is much easier to use than 'Refine by DATE' in MSA. Often, filter by year is specific enough and one exact date and time is too specific.

Respondent 2 A nice way to track down especially older emails, the ones where you vaguely know some words and a time period.

Respondent 3 Graphics are always nice. Very easy to zoom in on a specific time if you don't know the precise date and time.

Respondent 4 The time line gives a quick global insight in the communication frequency, and is thus helpful for a first quick orientation.

Q12. How helpful were the proximity queries of MSA? (example: lord..rings)

Respondent 1 3

Respondent 2 2

Respondent 3 4

Respondent 4 8

Q12.1. Please explain your choice

Respondent 1 I don't ever seem to use this option and I don't see how this could be helpful with my search queries.

Respondent 2 I did not find these queries useful. Two word queries gave me the same results. What counts as proximity between the words (4, 6, 12?) was not clear.

Respondent 3 Unhelpful is not the right term, but I didn't have any use for this type of search query. It is not clear to me what close by is in this case. Is it a specific amount of words? The same sentence? Can see it being helpful in some situations though, so it is a nice feature.

Respondent 4 Were helpful. A nice feature.

Q13. How helpful were the lexicographic range queries of MSA? (example: boaa–bozz)

Respondent 1 3

Respondent 2 2

Respondent 3 3

Respondent 4 8

Q13.1. Please explain your choice

Respondent 1 I don't ever seem to use this option and I don't see how this could be helpful with my search queries.

Respondent 2 These gave too many results (there is a whole lot of words between boaa and bozz) and if I tried to make the lexicographic query more specific I tended to fall back on a prefix search.

Respondent 3 Unhelpful is not the right term, but I didn't have any use for this type of search query. Don't see many situations where this could be useful.

Respondent 4 Also a nice feature in paricular cases.

Q14. How often did you re-index your inbox file with MSA? (How many times, or roughly at what interval)

Respondent 1 Every other day, so newest emails are added.

Respondent 2 Twice

Respondent 3 Once during the week.

Respondent 4 Twice a week.

Q15.1. TBS Quick Filters: What options did you use?

Respondent 1 Unread

Respondent 2 Unread;Starred

Respondent 3 Content

Respondent 4 Unread

Q15.2. MSA: What value did you use for Number of emails to index?

Respondent 1 2500

Respondent 2 [empty]

Respondent 3 Empty (so all)

Respondent 4 I left the field empty, because the mailbox contained less than 3000 emails.

Q15.3. MSA: Which preview fields did you have enabled?

Respondent 1 From;To;Date

Respondent 2 From;To;Date

Respondent 3 From;To;Date

Respondent 4 From;Date

Q15.4. MSA: Which facet boxes did you have enabled?

Respondent 1 From;Date

Respondent 2 From;Date

Respondent 3 From;Date

Respondent 4 From;To;Date

Q15.5. MSA: Which email address format did you use?

Respondent 1 Name

Respondent 2 Both

Respondent 3 Name

Respondent 4 Both

Q15.6. Please explain your choices

Respondent 1 My whole inbox is 5000+ emails, so I think indexing 2500 is enough for my searches. And I used the defaults for preview fields and facet boxes.

Respondent 2 Most of the defaults were fine. The email address format I set to Both, because it is way safer to see the address to reduce the risk of spoofing.

Respondent 3 Standard configuration and didn't feel the need to change them.

Respondent 4 Sometimes the email address doesn't mention the name of the writer of the email, but only his or her function or department.

Q16. Do you prefer the "search as you type" approach of MSA and TBS Quick Filters, or the "search when you press Enter" approach of TBS Global Search?

Respondent 1 I do not prefer one over the other

Respondent 2 I do not prefer one over the other

Respondent 3 I prefer the "search as you type" approach

Respondent 4 I prefer the "search when you press Enter" approach

Q16.1. Please explain your choice

Respondent 1 This difference does not seem to influence my searching behavior. I always type my complete querie as intended before I have a proper look at the results.

Respondent 2 I tend to still press Enter just out of habit

Respondent 3 Quicker.

Respondent 4 "Search when you press Enter" gives me a feeling of being more in control.

Q17. Do you prefer the automatic prefix search of MSA and TBS Quick Filters, or the manual wildcard (*) approach of TBS Global Search?

Respondent 1 I do not prefer one over the other

Respondent 2 I prefer the manual wildcard approach

Respondent 3 I do not prefer one over the other

Respondent 4 I do not prefer one over the other

Q17.1. Please explain your choice

Respondent 1 I don't see any difference in the results so I do not prefer one over the other.

Respondent 2 This follows the way I search everywhere, from documents on my computer, find in page on the web, in code, etc.

Respondent 3 Both had their pros and cons.

Respondent 4 It all depends on what kind of search work you are doing - quick and dirty or detailed precision search.

Q18. Do you think that a useful search mechanism is missing in TBS and MSA?

Respondent 1 No

Respondent 2 No

Respondent 3 Yes

Respondent 4 No

Q18.1. If yes, what is missing in TBS and MSA?

Respondent 1

Respondent 2

Respondent 3 Possibility to search in multiple languages at the same time. So automatic translator. Probably a bit niche.

Respondent 4 I wouldn't know.They are bothe quite complete for my purposes

Q19. Is there anything else you want to say?

Respondent 1

Respondent 2

Respondent 3

Respondent 4 In general I find TBS easier to understand and work with, but I admit that MSA has some very useful query options.

B maildir2mbox script

```
# maildir2mbox.nim
# convert a maildir folder into an mbox file
import std/os
import std/streams
import std/strutils

proc appendEmail(strmOut: FileStream, path: string) =
    strmOut.writeLine("From ")
    let strmIn = newFileStream(path, fmRead)
    var line = ""
    while strmIn.readLine(line):
        if line.startsWith("From "):
            strmOut.write(" ")
        strmOut.writeLine(line)
    strmIn.close

proc main =
    let maildir = paramStr(1)
    let strmOut = newFileStream(paramStr(2), fmWrite)
    var stack = @[maildir]
    while stack.len != 0:
        for x in walkDir(stack.pop):
            when not defined(release):
                stdout.write x
            case x.kind
            of pcDir: stack.add(x.path)
            else: strmOut.appendEmail(x.path)
    strmOut.close

case paramCount()
of 2: main()
else: echo "Usage: ", getAppFilename(), " <maildir> <out.mbox>"
```