Open-Source Software for Appraisal and Processing of Email at Scale

Christopher (Cal) Lee
School of Information and Library Science
University of North Carolina at Chapel Hill

Society of American Archivists Research Forum
August 5, 2020
Motivation - Selection/Appraisal

• Despite progress on various technologies to support data management and digital preservation, relatively little progress on software support for the core activities of selection and appraisal

• Selection/appraisal decisions are based on various patterns

• When patterns can be identified algorithmically, software can assist the process

• LAMs frequently want to take actions that reflect contextual relationships

• Timeline representations and visualizations can also provide useful, high-level views of materials
Motivation - Email

- About 50 years of email creation
- Hundreds of billions of messages generated every day
- Most has little long-term retention value, but some absolutely does
- Despite presence of numerous other modalities, email still deeply embedded in activities, serving as massive source of evidence and information
- Often found in collections and acquisitions with other types of materials

http://hci.stanford.edu/~jheer/projects/enron/v1/
Review, Appraisal, and Triage of Mail (RATOM)

• Funded by Andrew W. Mellon Foundation (2019-2020)

• Developing and repurposing software (including NLP and machine learning) for selection/appraisal in BitCurator environment with hooks and enhancements to TOMES output

• Support iterative processing - information discovered at various points in the processing workflow can support further selection, redaction or description actions

• Mapping of timestamp, entity, sensitive features and other elements across the tools

Ray Tomlinson
Implemented first email program on ARPANET. Credited with invention of first email system.
Team Members

Cal Lee
PI

Antoine De Torcy
Software Engineer

Camille Tyndall Watson
Co-PI

Jamie Patrick-Burns
Investigator

Eliscia Kinder
Project Manager

Kam Woods
Technical Lead (UNC)

Sangeeta Desai
Technical Lead (NC DAR)

Caktus Group
Additional Software Development
Scope of the project

The RATOM project has several core development goals designed to serve the needs of collecting institutions tasked with preparing email collections for public access:

- Development of an integrated Python library to simplify parsing and processing PST, OST, and mbox email formats
- Development of utilities to support entity identification and export reports suitable for conducting automated and human-directed redaction actions at scale
- Development of an interface allowing processing archivists to browse email collections and mark messages as suitable for retention
- Development of utilities to apply machine learning techniques (by training on annotated message collections and/or unsupervised) to recognize candidate materials for retention
Generating features that are:
facilitates understanding
changes in assessments of
materials over time.

libratom (reusable library)
Python library to parse and analyze PST, OST, and mbox email formats
Wraps functions from libpff, Python mailbox, and spaCy (NLP)
Email message content, header, attachment extraction; entity identification and classification
Engineered to scale with core count and keep memory use flat per-core

https://www.github.com/libratom/libratom

Generating features that are:
VERIFIABLE
REPRODUCIBLE
REUSABLE
facilitates understanding changes in assessments of materials over time.
RATOM tools - Iterative Processing Interface

Assist archivists in reviewing email materials for retention and/or release.

- Import of email accounts from PSTs and entity identification via libratom
- Creation of processing accounts associated with individual email users
- Interactive review and tagging of email messages within these accounts (e.g. “record”, “non-record”, “redact”)
- Export of selected messages as EML for retention or release

https://github.com/StateArchivesOfNorthCarolina/ratom-deploy
With the current CLI, we can load different models (including user trained models) on demand for tasks/languages.
**libratom commands**

**entities** command now provides more structured and responsive feedback when progress is requested (progress bars for both file and message scans), performance improvements via tuning of job distribution.

**model** command provides granular control of entity ident model(s) in use, including access to previously released models.

**report** command generates a fast report, populating the sqlite3 schema without entities (but optionally including message text and headers).

**emldump** provides a mechanism for generating EML files using JSON structured message id lists produced by the web app (may also be used standalone).
libratom output

Many updates as of 0.4.x...

Datetime stamps now extracted from message header and stored in message table

Option to include all message text (stripped of markup and inline attachments) and headers in message table

MIME types for attachments now recorded in attachment table; types are verified vs IANA listed content types and subtypes

Various fixes and improvements (additional detail in configuration, character encoding checks, etc)

See this chart in detail in the README at https://github.com/libratom/libratom
libratom processing the Enron (EDRM v1.3) corpus

EDRM v1.3 Enron Corpus: Approximately 54GB, and includes 191 files, containing 758,341 messages

PST internal directory structure and message count scan:
16-core Threadripper 2950X: 1 minute
32-core Threadripper 3970X: 30 seconds

Entity extraction from all 750K messages (spaCy en_core_web_sm model):
16-core Threadripper 2950X: 2 hrs
32-core Threadripper 3970X: 1 hr 15m

Memory usage is bounded for the spaCy configuration and number of processes. For 32 processes, accessible memory is ~1.6GB/process, resident memory is ~500MB/process on average.

In libratom 0.4.3, this run yields a 3.8GB sqlite3 file (including plaintext message bodies), containing 18,548,102 entity instances.
libratom processing the Jeb Bush corpus

Jeb Bush corpus: Approximately **7.2GB**, includes **11 files** containing **251,509 messages**

**PST internal directory structure and message count scan:**

16-core Threadripper 2950X: < 1 minute
32-core Threadripper 3970X: < 10 seconds

**Entity extraction from all 252K messages (spaCy en_core_web_sm model):**

16-core Threadripper 2950X: ~48 minutes
32-core Threadripper 3970X: ~30 minutes

Memory usage is bounded for the spaCy configuration and number of processes. For 32 processes, accessible memory is ~**1.6GB/process**, resident memory is ~**500MB/process** on average.

In libratom 0.4.3, this run yields a **798MB sqlite3 file** (including plaintext message bodies), containing **7,655,587 entity instances**.
libratom processing the Gov. Kaine (Library of Virginia) sample corpus

Kaine sample corpus: Approximately 12.3GB, includes 1 PST file containing 79,538 messages

PST internal directory structure, message count, and attachment scan:

16-core Threadripper 2950X: < 10 seconds
32-core Threadripper 3970X: < 5 seconds

Entity extraction from all 252K messages (spaCy en_core_web_sm model):

16-core Threadripper 2950X: ~24 minutes
32-core Threadripper 3970X: ~15 minutes

Memory usage is bounded for the spaCy configuration and number of processes. For 32 processes, accessible memory is ~1.6GB/process, resident memory is ~500MB/process on average.

In libratom 0.4.3, this run yields a 504MB sqlite3 file (including plaintext message bodies), containing 3,496,221 entity instances.
A simple exploration of the Kaine sample

We can quickly examine various slices of the db output using a few simple SQL queries...

Example 1:

Entity groups by type, ordered by count.

```sql
sqlite> select count(*), label_ from entity group by label_ order by count(*) DESC;
1288056|PERSON
686345|ORG
444841|DATE
362347|CARDINAL
257576|GPE
173187|TIME
60158|MONEY
32378|NORP
31311|ORDINAL
27683|FAC
27122|LOC
24327|PERCENT
23698|WORK_OF_ART
22738|PRODUCT
14511|LAW
11194|EVENT
8163|QUANTITY
586|LANGUAGE
sqlite>
```
A simple exploration of the Kaine sample

Example 2:

Individual text elements identified by spaCy as "PERSON" that appear more than 10,000 times

Note that the NLP processor will return full names as entities; it just happens that this particular group of entities was mentioned independently by first or last name a large number of times in the collection.

Formatting quirks can also produce the behavior - additional introspection into the materials and experiments with models other than the default model would be needed to determine whether this baseline performance could be significantly improved.

(As an example - the “Gail” and “Jaspen” elements here almost certainly refer to the same person, “Gail Jaspen”)
A simple exploration of the Kaine sample

Example 3:

Total number of attachments

Attachments by mime type, where there are more than 100 of that particular type

Note that given the total number of attachments this means there is a very long tail of additional attachment types...although some of these are variants of types that appear high in the list (Word, JPG, etc)

```sql
sqlite> select count(*) from attachment;
42783

sqlite> select count(*), mime_type from attachment group by mime_type having count(*) > 100 order by count(*) DESC;
23396|application/msword
4621|application/octet-stream
4355|application/vnd.ms-excel
3483|application/pdf
1833|image/jpeg
1685|image/gif
1363|application/vnd.ms-powerpoint
544|text/x-vcard
234|image/tiff
231|text/plain
179|text/html
136|application/msexcel
134|application/rtf
108|text/vcard
```
A simple exploration of the Kaine sample

Example 4:

We can use this db to explore data that might be problematic for processing further down the line. For example:

All attachments with identical names that appear in the collection more than 100 times

```
sqlite> select count(*), name from attachment group by name having count(*) > 100;
303|Document.pdf
108|IQFormatFile.txt
161|Scan001.PDF
729|image001.gif
1005|image001.jpg
152|image002.gif
157|image002.jpg
sqlite>
```
Releases and Code Quality

Updates and improvements as of 0.4.x:

Releases have been generated in tandem with tags on GitHub main branch, tracking all minor and patch updates (currently 0.4.3).

All releases automatically pushed to PyPI.

Travis CI runs now performed using Python 3.6, 3.7, and 3.8

Codebase now tracked with codeclimate to assess maintainability

Code coverage tracked via codecov (currently 95.47%) - effectively all core code is exercised by the test suite

Routine dependency tree checks via dependabot

Code vulnerability/security checks via bandit

Many others...
Notebooks and examples

**Libratom** is still in development, but as we add and test new features we’re making some of them available as Jupyter notebooks that you can try out.

These Jupyter notebooks can be run in any Jupyter Hub or Lab instance you choose, but for convenience we’ve configured them to run in a free hosted service - **mybinder** - in your web browser.

**Mybinder** can create a Jupyter Hub instance from any appropriately configured GitHub repository.

Click the “Launch Binder” badge in the ratom-notebooks repository to get started:

[https://github.com/libratom/ratom-notebooks](https://github.com/libratom/ratom-notebooks)

Note that mybinder is a free hosted service. Depending on the current load, it make take a few minutes for the project to start. Be patient!
RATOM tools - Iterative Processing Interface

Assist archivists in reviewing email materials for retention and/or release.

- Import of email accounts from PSTs and entity identification via libratom
- Creation of processing accounts associated with individual email users
- Interactive review and tagging of email messages within these accounts (e.g. “record”, “non-record”, “redact”)
- Export of selected messages as EML for retention or release

https://github.com/StateArchivesOfNorthCarolina/ratom-deploy
Accounts View

Accounts associated with imports of one or more imported PST files are displayed in the main interface.

Account processing indicates **Complete** when all entity identification and full-text indexing has finished.
Individual Account

Selecting an account displays an infinite-scroll view of individual messages associated with that account.

Green tags indicate entity classes identified during processing.

Status dropdown allows messages to be marked for retention or redaction (also appears in individual message view).
Message View

Messages are tagged during ingest using categories associated with entities identified in the body text.

(Note: this research dataset contains prior annotations, resulting in overtagging)
Tagging and Search

Selection by classification (e.g. record vs non-record) and date range.
Audit History

Audit histories for individual messages are retained, ensuring a clear record of initial processing actions and potential changes over time.

<table>
<thead>
<tr>
<th>PK</th>
<th>MESSAGE</th>
<th>PROCESSED</th>
<th>IS RECORD</th>
<th>DATE PROCESSED</th>
<th>UPDATED BY</th>
</tr>
</thead>
<tbody>
<tr>
<td>22035</td>
<td>Re: AGC Job Posting...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22034</td>
<td>Revised Draft...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22033</td>
<td>Re: Article...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22032</td>
<td>Re: New Revised Draft Answer - Ignore Pr...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22031</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22030</td>
<td>Lodi Storage...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22029</td>
<td>FW: CONFIRMATION: April 20, 2001 Executi...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22028</td>
<td>Houston...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22027</td>
<td>Re: Your Law Conference RSVP Form...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22026</td>
<td>Submit Your Law Conference RSVP Form...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22025</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22024</td>
<td>Re: FW: Draft Transwestern Response to F...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22023</td>
<td>Re: Additional Needs capacity...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22022</td>
<td>Re: Other update...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Project info, news, and blog posts:
https://ratom.web.unc.edu/
Core library:
https://github.com/libratom/libratom
Sample Jupyter notebooks:
https://github.com/libratom/ratom-notebooks
@RATOM_Project