Department of Computing Sciences

Assignment 1

Implementing `enron_search`: an Email Forensic Analysis Tool

Email continues to be an essential component of conducting business. This course-long project will introduce you to email forensics by having you write a tool that can analyze the Enron email data set. For instance, given a term to search, it will return the sender's email address and the date and time the message was sent by using the `From:` and `Date:` headers, respectively. **For this assignment, you will be allowed to work on teams of two or three students.** Some students may go solo if they prefer that, but no teams of more than three students please.

Instructions

1. Download the Enron email data set from https://www.cs.cmu.edu/~enron/.

   Be sure to get the May 7, 2015 version. The file will be named `enron_mail_20150507.tar.gz`.

2. Go to https://github.com/lintool/Enron2mbox and follow the instructions to convert the data to the mbox format.

3. For whatever language you plan to use for this assignment, find an mbox library.

   If you plan to use Python, check out the following links:

- https://docs.python.org/3/library/mailbox.html#mbox

  Take note that the `mbox` class is, because it is a subclass of the `Mailbox` class, an *iterator*, meaning you can use it in a `for` loop like this:

```
for message in my_mbox_class_instance:
```

- https://docs.python.org/3/library/mailbox.html#mailbox.mboxMessage

  `mailbox.mboxMessage` is a subclass of `mailbox.Message` which is a subclass of `email.message.Message`. You will need to understand the `email.message.Message` class's API for accessing the email's payload. Take

note that `email.message.Message` is **not** a subclass of `email.message.EmailMessage`.

4. Write a program that satisfies the description above and conforms to the following usage specifications:

```
enron_search term_search term [term ...]
```

`term`: A word to search for in the data set. The search will be case-insensitive, but exact, meaning neither fuzzy matching nor partial matching is performed. When more than one term is given, only emails with ALL terms in the body will be returned.

Your program should ignore duplicate terms and term order, so that the following are equivalent:

```
enron_search term_search the The THE money
enron_search term_search MONEY tHe monEY
enron_search term_search the money
```

The exclusion of fuzzy matching means that the term `cash` will not match the string `money`, although they are semantically similar. Exact matching (no partial matching) means `the` will not match the string `them`.

For each email with a message body (payload) that matches all the terms given by the user, you should capture and output the sender (using the `From:` header field) and the date the email was sent (using the `Date:` header field). Your program should number the results and display the total number of results found when the search completes. It is totally fine for your program to output both the sender's email address and the date/time sent in the same formats as they are stored in the email headers.

The following examples are notional (i.e., made up) and are **not** from the Enron data set. Lines starting with `$` are what the user enters into the command line. The other lines are the program's output.

```
$ enron_search term_search hide all the evidence
 1. Guy Incharge <incharge@enron.com> Mon, 18 Mar 1995 14:47:38
-0500
2. Peon Smith <psmith@enron.com> Tue, 19 Mar 1995 14:47:38 -0500
 3. Guy Incharge <incharge@enron.com> Wed, 20 Mar 1995 14:47:38
-0500
4. Peon Smith <psmith@enron.com> Thu, 21 Mar 1995 14:47:38 -0500
Results found: 4
```

5. Implement a new command to obtain all the emails sent and received by a given person:

    - `enron_search address_search last_name first_name`

    where `last_name` and `first_name` belong to the person you are trying to obtain the email addresses from. As a result, you will list the information of each obtained address as follows:

```
$ enron_search address_search carlos rubio-medrano
1. <carlos.rubiomedrano@tamucc.edu>
2. <crubiome@asu.edu>
Results found: 2
```

1. Implement a new command to obtain all the emails exchanged by two people, regardless of who initiated the communication in the first place:

    - `enron_search interaction_search address_1 address_2`

    where `address_1` and `address_2` identify the two people interacting with each other. The results should be displayed back to the screen as follows:

```
$ enron_search interaction_search crubiome@asu.edu dean@tamucc.edu
1. <crubiome@asu.edu> -> <dean@tamucc.edu> [Subject: Inquiry on Open Position] Mon, 18 Mar 2020 14:47:38 -0500
2. <dean@tamucc.edu> -> <crubiome@asu.edu> [Subject: Re: Inquiry on Open Position] Mon, 18 Mar 2020 14:57:38 -0500
Results found: 2
```

Implementation

You are free to use the programming language of your choice. However, **you are required to implement the command line interface** we discuss next, so we can properly test your code and we can correctly award you all the points you may deserve for your hard work. Please be advised that any deviations from the expected behavior of your code, e.g., it fails to compile or implement the requested command line interface correctly, will result in points being deducted from your grade.

Submission Instructions

Submit your source code as a **ZIP file** on the *Course Content/ Assignments/Assignment 1/Assignment 1* submission link located on our Course Blackboard Site. Also, submit a

separate README file containing the name of all the members of the team, TAMUCC ID, and a description of how your program works. To avoid confusions, **only one member of the team should submit the assignment**. Do NOT submit the Enron data set with your code!

Academic Integrity

All submissions to this assignment will be checked for plagiarism, and all detected cases will be promptly reported to TAMUCC authorities. Please direct any questions or clarifications regarding this assignment to the discussion forum labeled as *Assignment-1 Questions* within the class Blackboard site.