

Design Document

General overview of system with a small user guide.

Open command prompt and run the command `python3 phase1.py`

Follow instruction to create text files terms, emails, records, and dates

If input is not provided, default values are chosen for the user

Once the program is complete run the command `python3 phase2.py`

Follow instructions to create idx files te, re, da, and em

If input is not provided, default values are chosen for the user

Once the program is complete run the command `python3 phase3.py`

Input either 'output=full' or 'output=brief'

full output is the default output method

insert queries in the form of 'subj: gas'

for multiple search terms, enter the query in the form 'sub:gas subj: and'

you can also enter the query in the form 'term', in which case it returns result for where 'term' is in either subject or body

Detailed design of software

Phase 1:

Create terms/emails/dates/recs:

(process is similar for all, with slight variation noted with underlines)

Open a file to write the terms of each email into

Load xml file in chunks, with each chunk being one email record

For each email grab row id

Write row id into file

Traverse record until the <subj> tag

Split terms in record after <subj> tag to </subj> tag

Write each term into file with its corresponding row id

Phase 2:

Create re.idx/te.idx/em.idx/da.itx:

(process is similar for all, with slight variation noted with underlines)

Use linux sort command to sort records file, and get rid of duplicate values

Write the sorted records file into a temp file

Use the temp sorted file to create an appropriate temp file for db_load (key on line1 and data on line 2)

Use db_load to create a unique database for records file

Use command db_dump to create the re.idx file and unique database

Phase 3:

set mode to output=full

get input from user for mode

if mode is full do nothing

if mode is brief change mode to output=brief

get input for query

compute output for query

print output

let user input more queries

for multiple queries in an input, compute output for each query individually and output the intersecting values of each query

Testing strategy

Phase 1:

Manual compare of each file output with provided output example on eclass

Tested on both 1k and 10k file for efficiency

Phase 2:

Used db_load to compare output with input of each function

Tested on both 1k and 10 k for efficiency

Phase 3:

input: 'subj:gas'

input: 'subj:gas subj:and'

input: 'the'

group work break-down strategy.

Ishara Hettiarachchige:


Phase 1, phase 2, design doc

Matthew Braun:

Phase 3

Time spent: 7 hours

Method of coordination:

Online  git hub repository

4 In person meetups to check on progress, and submit files