MSCI 541 – Search Engines

Homework 1

Professor Mark Smucker

Matthew Erxleben

ID: 20889980

Date: September 27th, 2023

Problem 1:

In the article, the users are broken into buckets. However, if the buckets are recycled between experiments, then the next experiments will be biased due to the same users all experiencing the same carry over effect. This means that the same buckets of users cannot be recycled between experiments, as it will excel the carry over effects. Instead, the buckers should be randomized, however only a subset of buckets that won't affect the rest of them. This is called localized re-randomization, and I would employ this tactic whenever I am running experiments for my new search engine ranking algorithm. Another way I would avoid carry over effects, is by doing multiple different approaches for research and development (not just A/B experiments). I would utilize in-situ and lab studies to isolate the change, control the variables, with a small group of users. I would also use panels on larger groups of users to achieve long term statistics and demographics. Not relying on one way of obtaining data from users to improve my ranking algorithm will be key to avoiding the bad carry over effects, as seen in the article.

Problem 2:

Precision = # of relevant items found / total # of items retrieved

A)

Precision = 2 / 5

= 40%

B)

Precision = 2 / 5

= 40%

C)

The use of binary relevance and using a set-based retrieval measures can hide the true quality difference between the two rankings. Binary relevance is the concept that a result is either "relevant" or "not relevant". There is no in-between, a result cannot be somewhat relevant. It is binary, a 1 or a 0. These can hide the quality difference because there is no in-between or relevant or not relevant. And there is no measure of how relevant something is. This is also all up to the user, therefore it is difficult to measure as relevance is completely subjective to only that user's opinion. This causes something to be somewhat relevant, to be valued the same as something very relevant, just because they are both technically relevant according to the user. Set-based retrieval is when the user retrieves the results of their search, it is given to them in no particular order. The set is just the items that the search engine retrieved, without ranking the items in anyway. This can hide the quality due to a user putting bias on what they see first versus last, and preferring something based on the order that is retrieved. In rank-based retrieval, the

order of the items does matter. However, in set-based retrieval, users are not given results in any order and therefore this can cause users to miss more relevant items due to them not being ordered.

D)

Issue with binary relevance example:
I want to know about Shark Attacks in Cape Cod
Result 1: Shark Attacks: an in-depth look
Result 2: Cape Cod's most recent Shark Attack

Both of these results are relevant, due to both results involving shark attacks. However, result 2 is much more relevant than result, because it is almost exactly the same as my search. Due to binary relevance, both of these results are weighted equally as "relevant".

Issue with set-based retrieval example:

Search engine retrieves 100 items for user's search. Due to there being so many results, the user only pays attention to the first 50 results that they look at, and grazes over the last 50. However, there was a very relevant item in the last 50 items that the user looked at, but they didn't realize due to it being at the end of the results that they looked at. If this was rank-based, the ranking would likely put that very relevant item higher in the order so the user sees it first.

Problem 3:

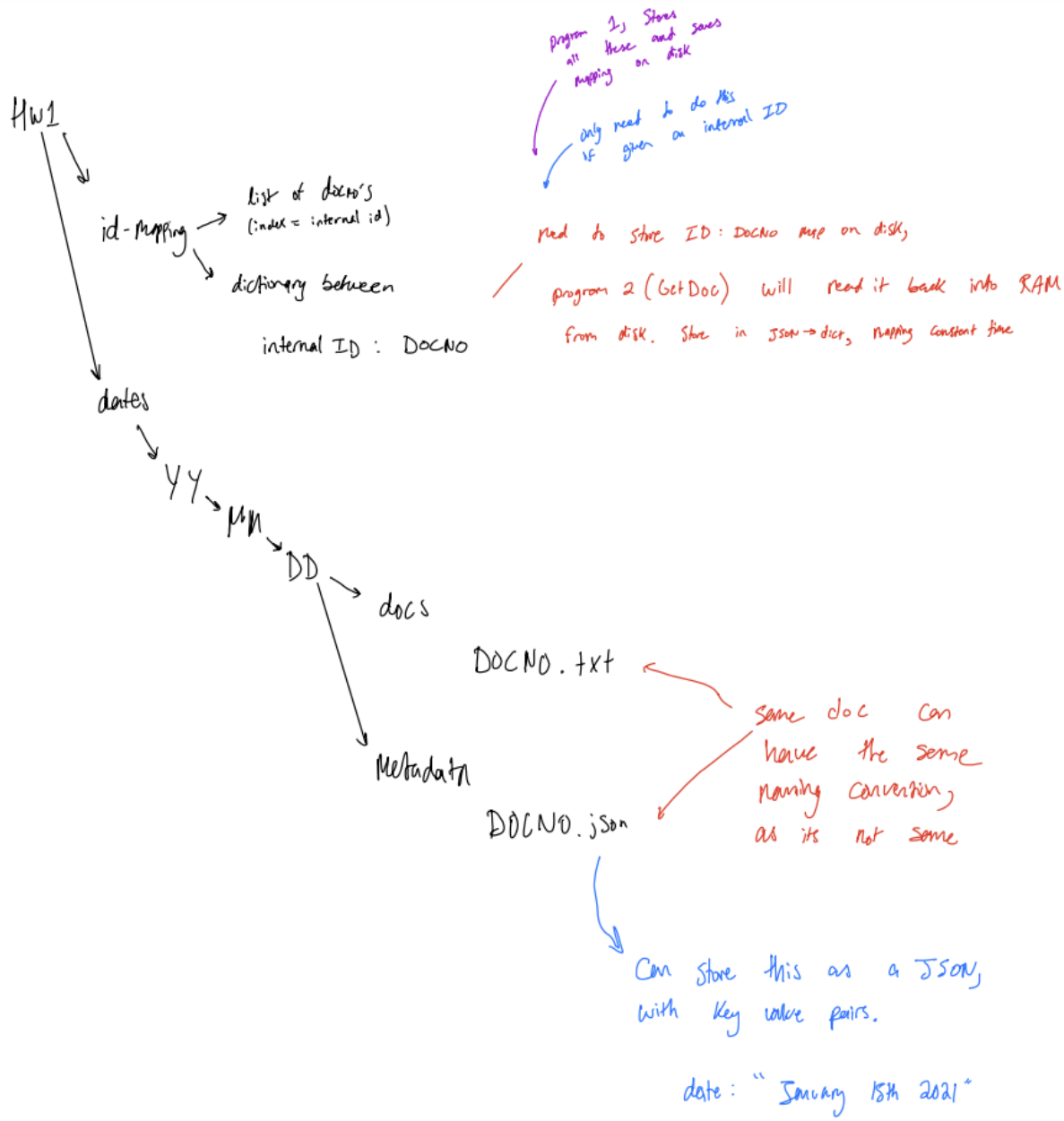
Due to this question being dropped (not marked) as announced by professor Mark Smucker on Campus Wire, this problem was not completed.

Problem 4:

a) Approach, Design, and Description

The first step I took towards approaching this problem was to try and understand on a high level what was needed for the two programs; IndexEngine and GetDoc. When looking at the problem, I discussed with fellow classmate Thomas Kleinknecht regarding the problem that was given to us and design choices. I planned out some pseudo code and diagrams to further approach the

problem. Here is a rough draft of my ideas when building out my logic:



First, I will look at IndexEngine's design and explain how I built it. I knew that IndexEngine needed to be in a file structure to store the documents and meta data. I used a /store folder, and then in that a /files/YY/MM/DD directory design to store what document is from that particular year, month, and day. In that DD folder I then have two separate folders; docs and meta_data. In the doc's folder, the raw document will be saved and is named its specific DOCNO.txt. In the meta_data folder, the meta data for that particular document is saved in the format:

- docno: LA010189-0018
- internal id: 76235
- date: January 1, 1989
- headline: OUTTAKES: MATERIAL MOL

This is saved in a Json file named its specific DOCNO.json. I utilized Json files for the meta data, because it is easy to transfer between Json files and dictionaries in Python when bringing the meta data between RAM and storing it on my disk.

Then if we move back to the /store folder, I have another folder called /id_map. In /id_map, I have two folders: /id_to_docno and /docno_to_id. /id_to_docno contains a file id_to_docno.txt, which is a text file containing every DOCNO, each on a new line. The order of where each DOCNO is represents its internal ID, and that is used when it is brough back into RAM later on in the program (by writing each line to an element in a list, and obtaining its index in the list, which gives us its internal id). /docno_to_id contains a json file docno_to_id.json, which is a json that maps the DOCNO : internal ID for every document.

These directories are created by my IndexEngine program using Pythons built-in os methods, and the store input from the user. IndexEngine then adds each raw document and its meta data to its respective YY/MM/DD based on its date (which is obtained from the 6 digits in DOCNO string after LA). The meta data is taken out of the raw doc using a for each loop and if statements depending the line (if headline, if docno) are used to then take the data out of the raw document. All the DOCNO's and internal IDs are stored in the id_to_docno and /docno_to_id folders to store the mapping on disk.

Now I will talk about the design of GetDoc. When passing a docno to GetDoc, the program will utilize the 6 digits in DOCNO string after LA to obtain the date of the article. This date is then used to traverse through the YY/MM/DD directory to obtain the raw document from the /docs folder using DOCNO.txt and obtain the meta data from the /meta_data folder using DOCNO.json. These two files are brought into RAM, the meta data json is turned to a dictionary, and the raw document text file is turned to a string. This is in constant time O(1).

When the user passes an internal ID to GetDoc, the program obtains the id_to_docno.txt file from the /id_mapping and turns each line into an element in a list in RAM. The internal ID given by the user is then used as an index in the list, to obtain the documents respective DOCNO. Now that the DOCNO, the same approach when given a DOCNO originally is used.

After the meta data and raw document are both borught into ram, the program displays the meta data, then the raw document in its output like:

docno: LA010189-0018

internal id: 76235

date: January 1, 1989

headline: OUTTAKES: MATERIAL MOLL

raw document:

<DOC>

… raw document contents …

</DOC>

b) Technology and Requirements:

I utilized the Python programming language to create these 2 programs.

Installation Requirements:

1. Please make sure Python is installed on your computer before running the program.
2. Clone repository on your device by entering this into your terminal: git clone https://github.com/UWaterloo-MSCI-541/msci-541-f23-hw1-matterxleben.git

In order to run these programs, please follow these instructions:

IndexEngine:

This program accepts two command line arguments:

1. a path to the latimes.gz file
2. a path to a directory where the documents and metadata will be stored.

For example, you would run IndexEngine from the command prompt / terminal / shell as:

python IndexEngine.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/raw-data/latimes.gz C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store

GetDoc:

The program accepts three command line arguments:

1. a path to the location of the documents and metadata store created by the first program (IndexEngine)
2. either the string "id" or the string "docno"
3. either the internal integer id of a document or a DOCNO

For example, you would run GetDoc from the command prompt / terminal / shell as:

python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store docno LA010189-0003

OR

python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store id 2

c) Testing Plan and Evidence of Functionality:

In order to show evidence that all programs work correctly, I will be looking at each program and seeing testing its functionality. I will provide evidence for the programs passing each test and showing its overall functionality with screenshots of my console (showing command line inputs, and what is outputted to the user).

IndexEngine:

Overall functionality:

Running this in the terminal:

python IndexEngine.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/raw-data/latimes.gz C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben> python IndexEngine.py
C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/raw-data/latimes.gz C:/Users/matth/OneDrive/Desktop/Univ
ersity/3B/MSCI541-Search-Engines/HW1/store
Created the directory: C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store
Completed storing all documents and metadata in the respective directory!
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben>
```

Creates the store folder:

« Users › matth › OneDrive › Desktop › University › 3B › MSCI541-Search-Engines › HW1

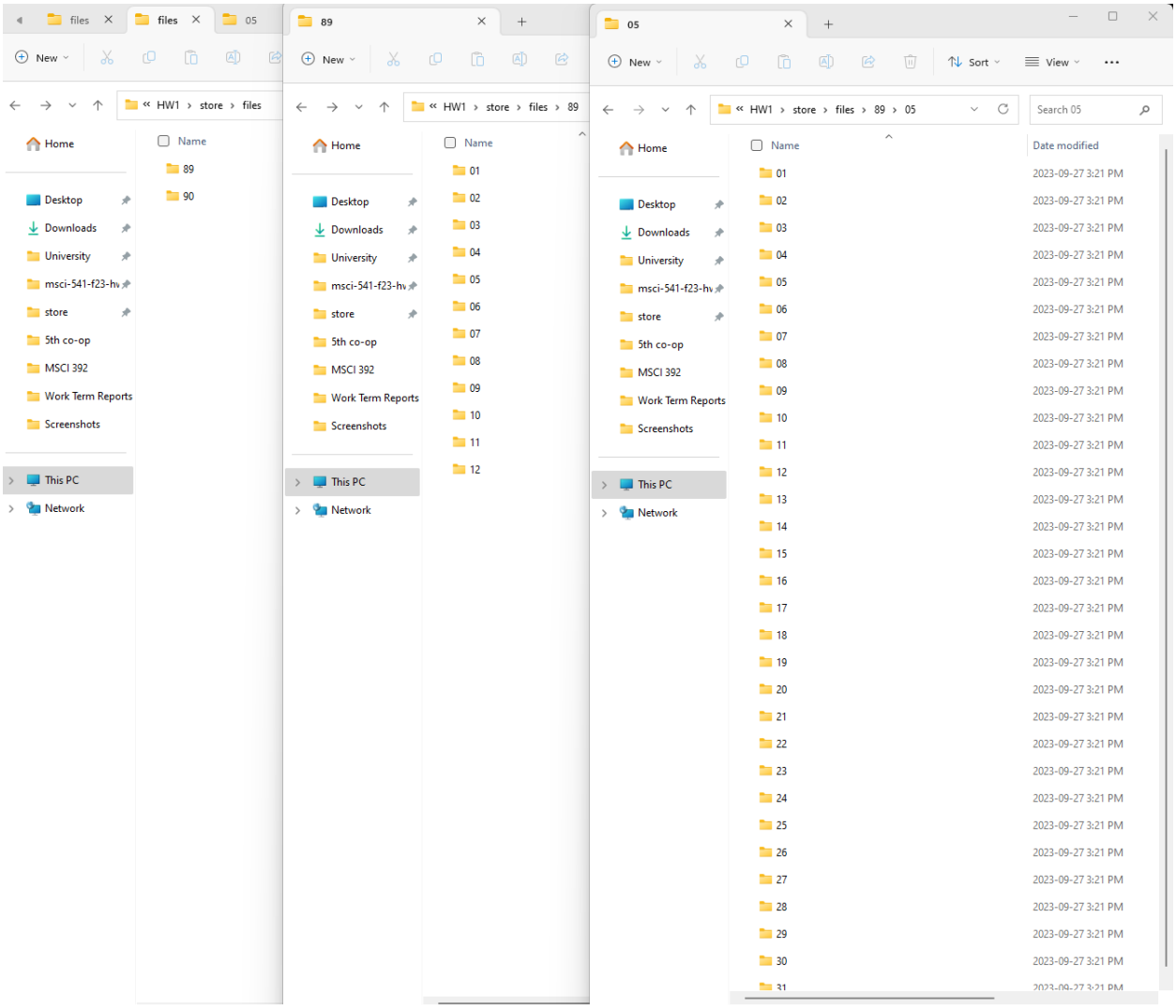
Name	Date modified	Type	Size
msci-541-f23-hw1-matterxleben	2023-09-27 2:51 AM	File folder	
raw-data	2023-09-26 2:20 PM	File folder	
store	2023-09-27 3:24 PM	File folder	

Creates the files (for docs and meta_data) and id_map folders:

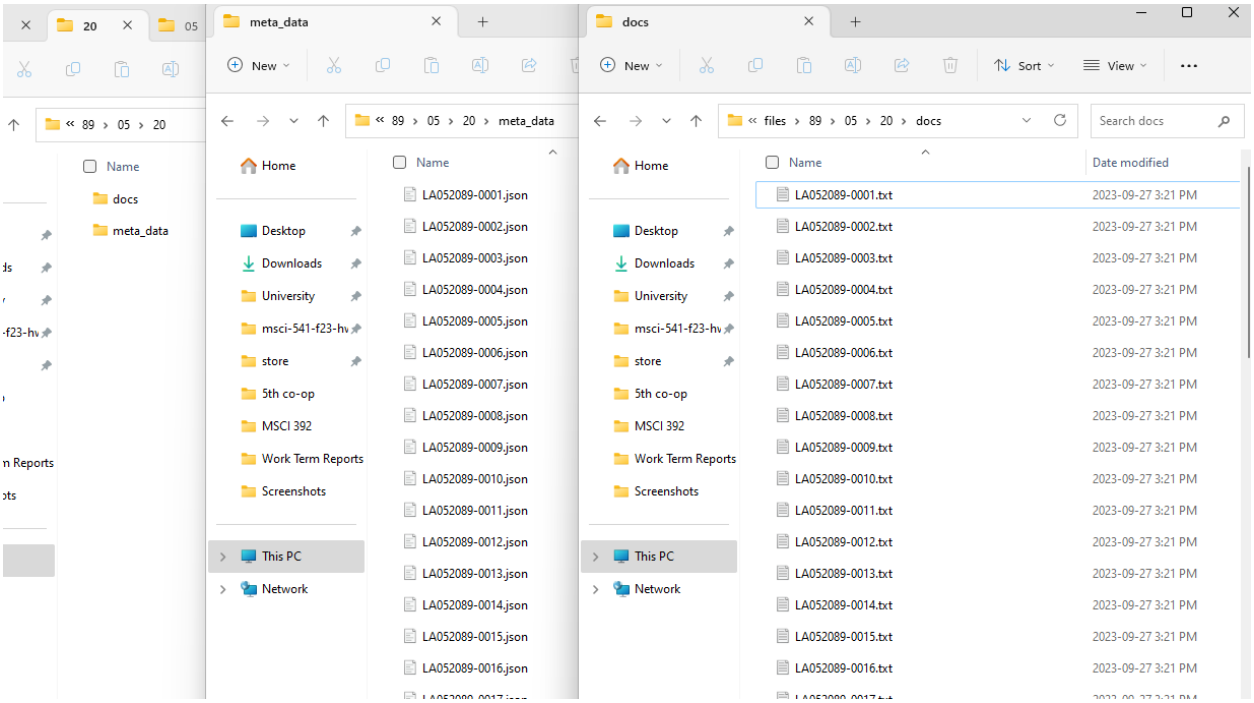
« OneDrive › Desktop › University › 3B › MSCI541-Search-Engines › HW1 › store ›

Name	Date modified	Type
files	2023-09-27 3:20 PM	File folder
id_map	2023-09-27 3:24 PM	File folder

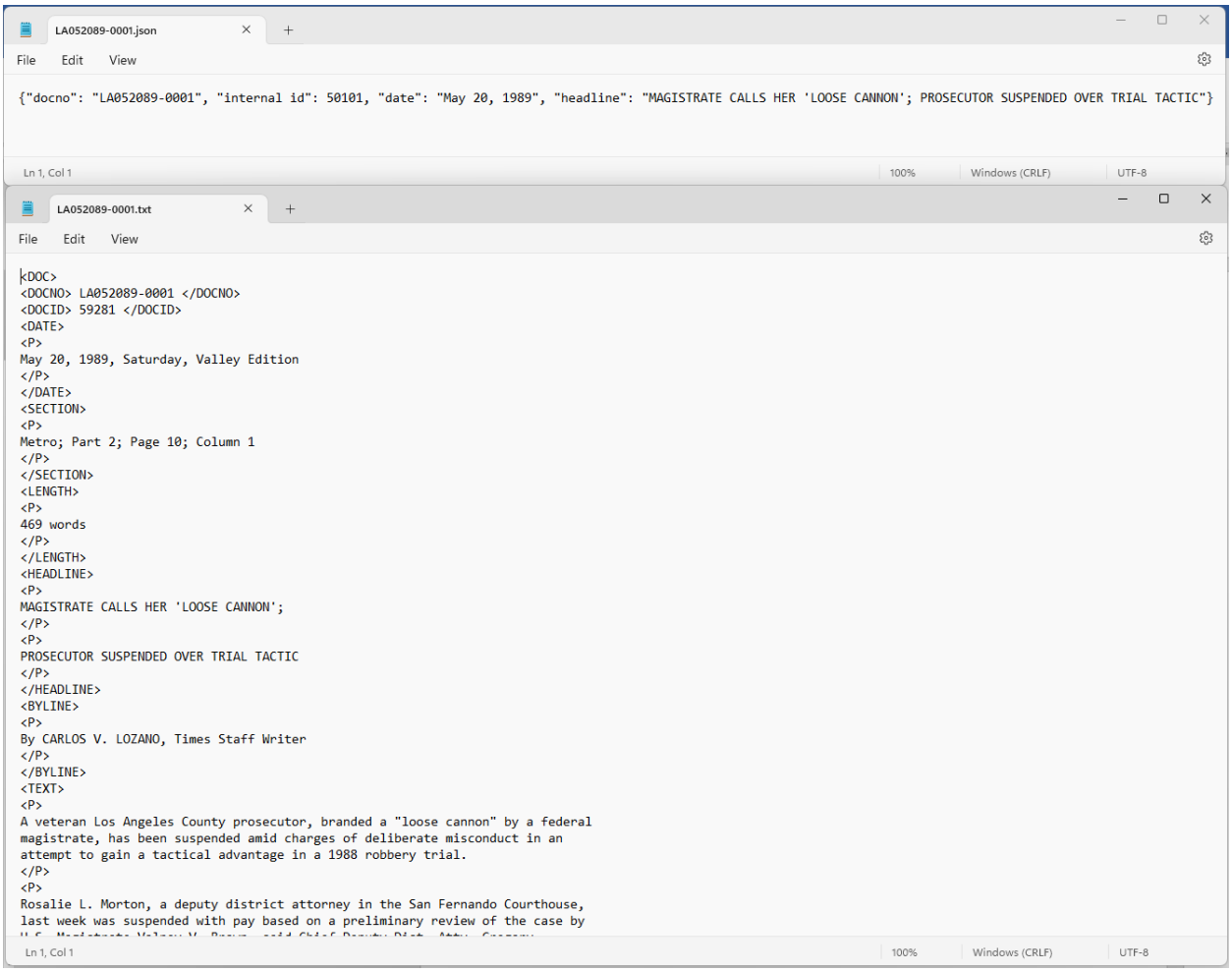
Creates YY/MM/DD directory:



In an example directory such as 89/05/20, the two folders “docs” and “meta_data” are created. Here is an example of the files that are stored in each folder:


meta_data
docs
New
Sort
View
« 89 › 05 › 20
« 89 › 05 › 20 › meta_data
« files › 89 › 05 › 20 › docs
Search docs
Home
Desktop
Downloads
University
msci-541-f23-hw
store
5th co-op
MSCI 392
Work Term Reports
Screenshots
This PC
Network
Name
Date modified
docs
meta_data
LA052089-0001.json
LA052089-0002.json
LA052089-0003.json
LA052089-0004.json
LA052089-0005.json
LA052089-0006.json
LA052089-0007.json
LA052089-0008.json
LA052089-0009.json
LA052089-0010.json
LA052089-0011.json
LA052089-0012.json
LA052089-0013.json
LA052089-0014.json
LA052089-0015.json
LA052089-0016.json
LA052089-0001.txt
LA052089-0002.txt
LA052089-0003.txt
LA052089-0004.txt
LA052089-0005.txt
LA052089-0006.txt
LA052089-0007.txt
LA052089-0008.txt
LA052089-0009.txt
LA052089-0010.txt
LA052089-0011.txt
LA052089-0012.txt
LA052089-0013.txt
LA052089-0014.txt
LA052089-0015.txt
LA052089-0016.txt
2023-09-27 3:21 PM

The docno.json file contains the documents meta data, and docno.txt file contains the raw document:



Edge Cases:

1. **No arguments supplied to program:**

Prints out the response: “”” This input does not meet the requirements for this program! The IndexEngine program's goal is to read the latimes.gz file and be able to store separately each document and its associated metadata. This program accepts two command line arguments:

1. a path to the latimes.gz file
2. a path to a directory where the documents and metadata will be stored.

For example, you would run IndexEngine from the command prompt / terminal / shell as:

python IndexEngine.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/raw-data/latimes.gz C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store “”””, and then exits the program.

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben> python IndexEngine.py

            This input does not meet the requirements for this program!
            The IndexEngine program's goal is to read the latimes.gz file and be able to store separately each document and its associated metadata.

            This program accepts two command line arguments:
                1. a path to the latimes.gz file
                2. a path to a directory where the documents and metadata will be stored.

            For example, you would run IndexEngine from the command prompt / terminal / shell as:
                python IndexEngine.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/raw-data/latimes.gz C:/Users/matth/OneDr
ive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store
```

2. **More than 2 arguments supplied to the program:**

Prints out the response: “”” This input does not meet the requirements for this program! The IndexEngine program's goal is to read the latimes.gz file and be able to store separately each document and its associated metadata. This program accepts two command line arguments:

1. a path to the latimes.gz file
2. a path to a directory where the documents and metadata will be stored.

For example, you would run IndexEngine from the command prompt / terminal / shell as:

python IndexEngine.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/raw-data/latimes.gz C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store “”””, and then exits the program.

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben> python IndexEngine.py 2 2 3 4 adasdasd sd

            This input does not meet the requirements for this program!
            The IndexEngine program's goal is to read the latimes.gz file and be able to store separately each document and its associated metadata.

            This program accepts two command line arguments:
                1. a path to the latimes.gz file
                2. a path to a directory where the documents and metadata will be stored.

            For example, you would run IndexEngine from the command prompt / terminal / shell as:
                python IndexEngine.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/raw-data/latimes.gz C:/Users/matth/OneDr
ive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store
```

3. **Latimes data file/directory does not exist:**

Prints out the response “This path does not exist! Please enter the correct path to the latimes data!”, and then exits the program

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben> python IndexEngine.py
 C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/raw-datadadsadadsaa/latimes.gz C:/Users/matth/OneDrive
/Desktop/University/3B/MSCI541-Search-Engine
This path does not exist! Please enter the correct path to the latimes data!
```

4. **Store directory already exists:**

Prints out the response: “This storing directory already exists! Please enter a new storing directory that does not already exist (or delete the directory off your DISK that your are looking to store to) and rerun this program!”, and then exits the program

```
C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/raw-data/latimes.gz C:/Users/matth/OneDrive/Desktop/Univ
ersity/3B/MSCI541-Search-Engines/HW1/storeOneDrive\Desktop\University\3B\MSCI541-Search-Engines/HW1/storeOneDrive\-Engin-Engin-En
-Engines/HW1/storeOneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben>
This storing directory already exists! Please enter a new storing directory that does not already exist (or delete the directory
off your DISK that your are looking to store to) and rerun this program!
```

GetDoc:

Overall functionality:

There are two ways that GetDoc can retrieve a raw document and its meta data:

1. Retrieval by Internal ID
2. Retrieval by DOCNO

Therefore, lets test these two retrieval strategies:

Retrieval by Internal ID:

Running this in the terminal:

python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store id 2

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben> python GetDoc.py C:/Users/matth/OneDrive
/Desktop/University/3B/MSCI541-Search-Engines/HW1/store id 2
docno: LA010189-0003
internal id: 2
date: January 1, 1989
headline: PERUVIAN MEMORIES AND THE 'SHINING PATH'; TUNGSTEN A NOVEL BY CESAR VALLEJO; TRANSLATED BY ROBERT MEZEY; FOREWORD BY KEVIN J. O'CONNOR (S
YRACUSE UNIVERSITY PRESS: $19.95; 168 PP.; 0-8156-0226-X)
raw document:
<DOC>
<DOCNO> LA010189-0003 </DOCNO>
<DOCID> 3 </DOCID>
<DATE>
<P>
January 1, 1989, Sunday, Home Edition
</P>
</DATE>
<SECTION>
<P>
Book Review; Page 2; Book Review Desk
</P>
</SECTION>
<LENGTH>
<P>
1194 words
</P>
</LENGTH>
<HEADLINE>
<P>
PERUVIAN MEMORIES AND THE 'SHINING PATH';
</P>
<P>
TUNGSTEN A NOVEL BY CESAR VALLEJO; TRANSLATED BY ROBERT MEZEY; FOREWORD BY
KEVIN J. O'CONNOR (SYRACUSE UNIVERSITY PRESS: $19.95; 168 PP.; 0-8156-0226-X)
</P>
</HEADLINE>
<BYLINE>
<P>
By Edith Grossman, Grossman is a critic and translator of Latin American
literature. She teaches at Dominican College in New York State, is the author
of The Antipoetry of Nicanor Parra and recently translated Gabriel Garcia
Marquez's Love in the Time of Cholera.
</P>
</BYLINE>
<TEXT>
```

Raw document continues until the </DOC> tag (end of document):

```
Last and certainly least, the third section of the book is a kind of postscript
to the Colca massacre. Vallejo allows his indignation and passion to turn into
mechanical ideology: Servando Huanca "duckspeaks" a mercifully brief exercise
in orthodoxy, proclaiming that world revolution is under way, that it will be
led by a militant proletariat inspired by Lenin and not by bourgeois
"intellectuals," that it will sweep away the ruling classes in Peru. Vallejo
even permits the novel to end with this ponderously symbolic sentence:
"Outside, the wind was rising, portending storm."
</P>
<P>
Kevin J. O'Connor has written a thoughtful and informative foreword, Robert
Mezey has done a fine job of translating Vallejo's often quirky Spanish, and
both of them deserve our gratitude for their sensitivity and skill in bringing
the work to the attention of an English-language audience. I am sorry that I
cannot say as much for the editing. The typographical errors are frequent and
unforgivable, and there is no indication as one reads that Mezey has provided
important end notes to the text. This is a disservice to the reader; Vallejo
and his translator deserve better.
</P>
</TEXT>
<TYPE>
<P>
Book Review
</P>
</TYPE>
</DOC>
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben>
```

As we can see, the output format is:

docno: LA010189-0003

internal id: 2

date: January 1, 1989

headline: PERUVIAN MEMORIES AND THE 'SHINING PATH'; TUNGSTEN A NOVEL BY CESAR VALLEJO; TRANSLATED BY ROBERT MEZEY; FOREWORD BY KEVIN J. O'CONNOR (SYRACUSE UNIVERSITY PRESS: $19.95; 168 PP.; 0-8156-0226-X)

raw document:

<DOC> … </DOC>

Retrieval by DOCNO:

To obtain the same document as we tested for in “retrieval by Internal ID” above, we can use the DOCNO: LA010189-0003

Running this in the terminal:

python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store docno LA010189-0003

```
/Desktop/University/3B/MSCI541-Search-Engines/HW1/store docno LA010189-0003
docno: LA010189-0003
internal id: 2
date: January 1, 1989
headline: PERUVIAN MEMORIES AND THE 'SHINING PATH'; TUNGSTEN A NOVEL BY CESAR VALLEJO; TRANSLATED BY ROBERT MEZEY; FOREWORD BY KEVIN J. O'CONNOR (S
YRACUSE UNIVERSITY PRESS: $19.95; 168 PP.; 0-8156-0226-X)
raw document:
<DOC>
<DOCNO> LA010189-0003 </DOCNO>
<DOCID> 3 </DOCID>
<DATE>
<P>
January 1, 1989, Sunday, Home Edition
</P>
</DATE>
<SECTION>
<P>
Book Review; Page 2; Book Review Desk
</P>
</SECTION>
<LENGTH>
<P>
1194 words
```

Raw document continues until the </DOC> tag (end of document):

```
to the Colca massacre. Vallejo allows his indignation and passion to turn into
mechanical ideology: Servando Huanca "duckspeaks" a mercifully brief exercise
in orthodoxy, proclaiming that world revolution is under way, that it will be
led by a militant proletariat inspired by Lenin and not by bourgeois
"intellectuals," that it will sweep away the ruling classes in Peru. Vallejo
even permits the novel to end with this ponderously symbolic sentence:
"Outside, the wind was rising, portending storm."
</P>
<P>
Kevin J. O'Connor has written a thoughtful and informative foreword, Robert
Mezey has done a fine job of translating Vallejo's often quirky Spanish, and
both of them deserve our gratitude for their sensitivity and skill in bringing
the work to the attention of an English-language audience. I am sorry that I
cannot say as much for the editing. The typographical errors are frequent and
unforgivable, and there is no indication as one reads that Mezey has provided
important end notes to the text. This is a disservice to the reader; Vallejo
and his translator deserve better.
</P>
</TEXT>
<TYPE>
<P>
Book Review
</P>
</TYPE>
</DOC>
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben>
```

As we can see, the output format is:

docno: LA010189-0003

internal id: 2

date: January 1, 1989

headline: PERUVIAN MEMORIES AND THE 'SHINING PATH'; TUNGSTEN A NOVEL BY CESAR VALLEJO; TRANSLATED BY ROBERT MEZEY; FOREWORD BY KEVIN J. O'CONNOR (SYRACUSE UNIVERSITY PRESS: $19.95; 168 PP.; 0-8156-0226-X)

raw document:

<DOC> … </DOC>

When comparing the “retrieval by Internal ID” and “retrieval by DOCNO” for the same document, we get the same output results! Therefore, this shows that both methods of obtaining documents are functional.

Edge Cases:

1. **No arguments supplied to program:**

Prints out the response: “”” This input does not meet the requirements for this program! The GetDoc program's goal is to efficiently retrieve a document and its metadata, based on inputs from the user.

The program accepts three command line arguments:

1. a path to the location of the documents and metadata store created by the first program (IndexEngine)
2. either the string "id" or the string "docno"
3. either the internal integer id of a document or a DOCNO

For example, you would run GetDoc from the command prompt / terminal / shell as:

python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store docno LA010189-0003

OR

python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store id 2 “”””, and then exits the program.

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben> python GetDoc.py

            This input does not meet the requirements for this program!
            The GetDoc program's goal is to efficiently retrieve a document and its metadata, based on inputs from the user.

            The program accepts three command line arguments:
                1. a path to the location of the documents and metadata store created by the first program (IndexEngine)
                2. either the string "id" or the string "docno"
                3. either the internal integer id of a document or a DOCNO

            For example, you would run GetDoc from the command prompt / terminal / shell as:
                python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store docno LA010189-0003
            OR
                python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store id 2
```

2. **More than 3 arguments supplied to program:**

Prints out the response: “”” This input does not meet the requirements for this program! The GetDoc program's goal is to efficiently retrieve a document and its metadata, based on inputs from the user.

The program accepts three command line arguments:

1. a path to the location of the documents and metadata store created by the first program (IndexEngine)
2. either the string "id" or the string "docno"

3. either the internal integer id of a document or a DOCNO

For example, you would run GetDoc from the command prompt / terminal / shell as:

python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store docno LA010189-0003

OR

python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store id 2 “””, and then exits the program.

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben> python GetDoc.py C:/Users/matth/OneDrive
/Desktop/University/3B/MSCI541-Search-Engines/HW1/store docno LA010189-0003 44 342 23 fkkfkfkf 333

            This input does not meet the requirements for this program!
            The GetDoc program's goal is to efficiently retrieve a document and its metadata, based on inputs from the user.

            The program accepts three command line arguments:
                1. a path to the location of the documents and metadata store created by the first program (IndexEngine)
                2. either the string "id" or the string "docno"
                3. either the internal integer id of a document or a DOCNO

            For example, you would run GetDoc from the command prompt / terminal / shell as:
                python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store docno LA010189-0003
            OR
                python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW1/store id 2
```

3. User does not enter “id” or “docno” for the second argument:

Prints out the response: “This input does not meet the requirements for this program! Please supply either "id" or "docno" as the second argument to the program on the command line.”, then exits the program.

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben> python GetDoc.py C:/Users/matth/OneDrive
/Desktop/University/3B/MSCI541-Search-Engines/HW1/store asdf LA010189-0003
This input does not meet the requirements for this program! Please supply either "id" or "docno" as the second argument to the program on the comma
nd line.
```

4. Meta data path / docs path does not exist or is incorrect:

Prints out the response: "This path does not exist! Please enter the correct path to the documents and meta data!", and exits the program

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben> python GetDoc.py C:/Users/matth/OneDrive
/Desktop/University/3B/MSCI541-Search-Engines/HW1/stdasdasd docno LA010189-0003
This path does not exist! Please enter the correct path to the documents and meta data!
```

5. DOCNO does not exist or is incorrect:

Prints out the response: "This path does not exist! Please enter the correct path to the documents and meta data!", and exits the program. (this is okay as the docno is what is used for the Path!)

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben> python GetDoc.py C:/Users/matth/OneDrive
/Desktop/University/3B/MSCI541-Search-Engines/HW1/store docno LA010189-0002783892390320932903
This path does not exist! Please enter the correct path to the documents and meta data!
```

6. Internal ID does not exist or is incorrect:

Prints out the response: “This Internal ID does not exist! Please enter a correct Internal ID for a document!”, and exits the program.

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW1\msci-541-f23-hw1-matterxleben> python GetDoc.py C:/Users/matth/OneDrive
/Desktop/University/3B/MSCI541-Search-Engines/HW1/store id 1319000
This Internal ID does not exist! Please enter a correct Internal ID for a document!
```