Mid-Term Practice Portion

Problem-1: User interaction with an IR system

Pick a site with a good number of text, html, or pdf documents. Crawl it using wget, collecting 100-1000 documents. You may have to do some trial-and-error to get to this number. Make sure you don't keep your crawler running for too long. You will be penalized for having fewer than 100 or more than 1000 indexed documents. Report your wget command. [5 points]

```
# This is my wget command that I used

# wget -r paulgraham.com
```

Index this collection using Pyterrier. Make sure you pick appropriate meta tags that may be useful for retrieval later. Report the index statistics. [5 points]

```
import pyterrier as pt
if not pt.started():
  pt.init()
```

```
C:\Users\Fortu\AppData\Local\Temp\ipykernel_9280\1742394333.py:2: DeprecationWarning: Call to
deprecated function (or staticmethod) started. (use pt.java.started() instead) -- Deprecated
since version 0.11.0.
  if not pt.started():
```

```
files = pt.io.find_files('c:/Users/Fortu/Downloads/Aut2024/Info 376/paulgraham.com')
indexer = pt.FilesIndexer('c:/Users/Fortu/Downloads/Aut2024/Info 376/midterm_practice/webcrawler_
                          verbose=True, blocks=False,
                          meta={"docno":20,"filename":1024,"title":1024},meta_tags={"title":"titl
                          overwrite=True)
indexref = indexer.index(files)
index = pt.IndexFactory.of(indexref)
print(index.getCollectionStatistics().toString())
```

```
16:24:03.828 [main] ERROR org.terrier.structures.indexing.Indexer -- Could not rename index
java.io.IOException: Rename of index structure file 'c:/Users/Fortu/Downloads/Aut2024/Info
376/midterm_practice/webcrawler_index/data_1.direct.bf' (exists) to
'c:/Users/Fortu/Downloads/Aut2024/Info 376/midterm_practice/webcrawler_index/data.direct.bf'
(exists) failed - likely that source file is still open. Possible indexing bug?
    at org.terrier.structures.IndexUtil.renameIndex(IndexUtil.java:379)
    at org.terrier.structures.indexing.Indexer.index(Indexer.java:388)
Number of documents: 384
Number of terms: 12539
Number of postings: 160977
Number of fields: 0
Number of tokens: 412945
Field names: []
Positions:   false
```

Create a client-side interface in PHP/HTML by making appropriate modifications to what was provided in the class. Make sure that in addition to the search box and button, it has some description of what the underlying collection is and offer some suggestions for what one could search for. [5 points]

You can access the search interface using the following link: [Search Interface](http://is-searchrec.ischool.uw.edu/swas/search.php)

(http://is-searchrec.ischool.uw.edu/swas/search.php)

```
import pandas as pd
queries = pd.DataFrame([["q1","addiction"], ["q2","life"], ["q3", "age"], ["q4", "USA"]], columns
queries
```

	qid	query
0	q1	addiction
1	q2	life
2	q3	age
3	q4	USA

```
index = pt.IndexFactory.of("c:/Users/Fortu/Downloads/Aut2024/Info 376/midterm_practice/webcrawler_
BM = pt.BatchRetrieve(index, wmodel="BM25")
BM.transform(queries)
```

```
C:\Users\Fortu\AppData\Local\Temp\ipykernel_9280\1970710156.py:2: DeprecationWarning: Call to
deprecated class BatchRetrieve. (use pt.terrier.Retriever() instead) -- Deprecated since version
0.11.0.
  BM = pt.BatchRetrieve(index, wmodel="BM25")
```

	qid	docid	docno	rank	score	query
0	q1	15	d16	0	9.957912	addiction
1	q1	87	d88	1	8.445748	addiction
2	q1	176	d177	2	8.236193	addiction
3	q1	352	d353	3	8.205934	addiction
4	q1	133	d134	4	6.743001	addiction
...	...	...	...	...	...	...
180	q3	187	d188	59	1.338348	age
181	q3	316	d317	60	1.030054	age

	qid	docid	docno	rank	score	query
182	q4	347	d348	0	10.398220	USA
183	q4	37	d38	1	7.369159	USA
184	q4	323	d324	2	5.813612	USA

185 rows × 6 columns

Kill any existing server you have running in the background. Now edit and start the server script and have it listen to your assigned port. When one enters a query on your search page, it should display ranked results from your index. Submit the URL for this search page. [10 points]

```
# my assigned port (10014) was taken so I used 10001
```

Problem-2: Indexing to evaluation

Use the LA Times data in TREC format available from here to index using Pyterrier. Make sure you index with tags 'DOCNO' and 'HEADLINE'. Report index statistics. [10 points]

```
files = pt.io.find_files('c:/Users/Fortu/Downloads/Aut2024/Info 376/latimes')
indexer = pt.TRECCollectionIndexer('c:/Users/Fortu/Downloads/Aut2024/Info 376/midterm_practice/la
              verbose=True, blocks=False,
              meta={"docno":20,"filename":1024,"headline":1024},meta_tags={"headline":"headline"}
              overwrite=True)
indexref = indexer.index(files)
index = pt.IndexFactory.of(indexref)
print(index.getCollectionStatistics().toString())
```

```
2files [01:46, 53.07s/files]

Number of documents: 131896
Number of terms: 188927
Number of postings: 24807798
Number of fields: 0
Number of tokens: 40507829
Field names: []
Positions:   false
```

Show how you can retrieve the docno for a document in that collection by providing a docid (e.g., 0, 1, 2) with a couple of examples. Report your Pyterrier commands and outcomes. [5 points]

```
meta = index.getMetaIndex()
docid = 0
docno = meta.getItem("docno", docid)
print(f"Docid: {docid}, Docno: {docno}")

docid = 1
```

```
docno = meta.getItem("docno", docid)
print(f"Docid: {docid}, Docno: {docno}")
```

```
Docid: 0, Docno: LA010189-0001
Docid: 1, Docno: LA010189-0002
```

Prepare a query file or a dataframe for batch processing. This file should be in TREC format and should contain the following queries: “curbing population growth” (q435), “railway accidents” (q436), “inventions, scientific discoveries” (q439), “ship losses” (q448), “King Hussein, peace” (q450). The numbers in parentheses are query numbers. Show your query file or dataframe. [5 points]

```
queries_trec = pd.DataFrame([
  ["q435", "curbing population growth"],
  ["q436", "railway accidents"],
  ["q439", "inventions, scientific discoveries"],
  ["q448", "ship losses"],
  ["q450", "King Hussein, peace"]
], columns=["qid", "query"])

queries_trec
```

	qid	query
0	q435	curbing population growth
1	q436	railway accidents
2	q439	inventions, scientific discoveries
3	q448	ship losses
4	q450	King Hussein, peace

Perform retrieval using three different retrieval models/methods. Report the ranked results, up to 10, for each of these. [10 points]

```
index = pt.IndexFactory.of("c:/Users/Fortu/Downloads/Aut2024/Info 376/midterm_practice/la_times_i
BM = pt.BatchRetrieve(index, wmodel="BM25")
TF_IDF = pt.BatchRetrieve(index, wmodel="TF_IDF")
Hiem = pt.BatchRetrieve(index, wmodel="Hiemstra_LM")
```

```
C:\Users\Fortu\AppData\Local\Temp\ipykernel_9280\3012119167.py:2: DeprecationWarning: Call to
deprecated class BatchRetrieve. (use pt.terrier.Retriever() instead) -- Deprecated since version
0.11.0.
  BM = pt.BatchRetrieve(index, wmodel="BM25")
C:\Users\Fortu\AppData\Local\Temp\ipykernel_9280\3012119167.py:3: DeprecationWarning: Call to
deprecated class BatchRetrieve. (use pt.terrier.Retriever() instead) -- Deprecated since version
0.11.0.
  TF_IDF = pt.BatchRetrieve(index, wmodel="TF_IDF")
C:\Users\Fortu\AppData\Local\Temp\ipykernel_9280\3012119167.py:4: DeprecationWarning: Call to
deprecated class BatchRetrieve. (use pt.terrier.Retriever() instead) -- Deprecated since version
```

```
0.11.0.
  Hiem = pt.BatchRetrieve(index, wmodel="Hiemstra_LM")
```

```
BM_values = BM.transform(queries_trec)
TF_IDF_values = TF_IDF.transform(queries_trec)
Hiem_values = Hiem.transform(queries_trec)
```

```
print(BM_values.head(10))
```

```
    qid   docid          docno  rank      score                    query
0  q435  126171  LA121589-0139     0  25.390996  curbing population growth
1  q435   83083  LA082189-0092     1  23.927200  curbing population growth
2  q435   48392  LA051590-0008     2  23.758792  curbing population growth
3  q435  107074  LA102689-0071     3  23.073475  curbing population growth
4  q435   31179  LA032990-0138     4  22.825540  curbing population growth
5  q435  122235  LA120589-0035     5  22.547941  curbing population growth
6  q435   46844  LA051189-0089     6  21.957725  curbing population growth
7  q435   89431  LA090890-0109     7  21.014325  curbing population growth
8  q435    2032  LA010790-0036     8  20.837134  curbing population growth
9  q435   44580  LA050490-0199     9  20.675308  curbing population growth
```

```
print(TF_IDF_values.head(10))
```

```
    qid   docid          docno  rank      score                    query
0  q435  126171  LA121589-0139     0  14.082744  curbing population growth
1  q435   83083  LA082189-0092     1  13.269034  curbing population growth
2  q435   48392  LA051590-0008     2  13.174410  curbing population growth
3  q435  107074  LA102689-0071     3  12.799793  curbing population growth
4  q435   31179  LA032990-0138     4  12.655082  curbing population growth
5  q435  122235  LA120589-0035     5  12.527306  curbing population growth
6  q435   46844  LA051189-0089     6  12.206764  curbing population growth
7  q435   89431  LA090890-0109     7  11.647640  curbing population growth
8  q435    2032  LA010790-0036     8  11.587420  curbing population growth
9  q435   44580  LA050490-0199     9  11.474252  curbing population growth
```

```
print(Hiem_values.head(10))
```

```
    qid   docid          docno  rank      score                    query
0  q435  126171  LA121589-0139     0  15.093337  curbing population growth
1  q435   83083  LA082189-0092     1  13.261315  curbing population growth
2  q435   48392  LA051590-0008     2  13.008375  curbing population growth
3  q435  122235  LA120589-0035     3  11.152714  curbing population growth
4  q435   89431  LA090890-0109     4  10.840118  curbing population growth
5  q435  107074  LA102689-0071     5  10.820350  curbing population growth
6  q435   46844  LA051189-0089     6  10.618389  curbing population growth
7  q435   31179  LA032990-0138     7  10.535393  curbing population growth
8  q435    2032  LA010790-0036     8  10.483613  curbing population growth
9  q435  131639  LA123189-0076     9  10.285353  curbing population growth
```

Use the provided qrels document to perform evaluation on these results. Report MAP, R-precision, and reciprocal rank values. Provide your brief thoughts on how well your retrieval models are doing. [15 points]

```python
from pyterrier.measures import *

# Load the qrels file
qrels = pd.read_csv('C:/Users/Fortu/Downloads/Aut2024/Info 376/qrels.csv')

# Perform evaluation
results = pt.Experiment(
    [BM, TF_IDF, Hiem],
    queries_trec,
    qrels,
    ["map", "recip_rank", "Rprec"]
)

print(results)
```

```
                         name       map  recip_rank     Rprec
0          TerrierRetr(BM25)  0.021449    0.415009  0.044059
1        TerrierRetr(TF_IDF)  0.021637    0.414996  0.044742
2  TerrierRetr(Hiemstra_LM)  0.023595    0.425238  0.042602
```

Evaluation of Retrieval Models

The evaluation of the three retrieval models (BM25, TF_IDF, and Hiemstra_LM) using the metrics MAP, Reciprocal Rank, and R-precision provides insights into their performance:

- **BM25**:
    - **MAP**: 0.021449
    - **Reciprocal Rank**: 0.415009
    - **R-precision**: 0.044059
- **TF_IDF**:
    - **MAP**: 0.021637
    - **Reciprocal Rank**: 0.414996
    - **R-precision**: 0.044742
- **Hiemstra_LM**:
    - **MAP**: 0.023595
    - **Reciprocal Rank**: 0.425238
    - **R-precision**: 0.042602

Observations:

1. **MAP (Mean Average Precision)**: Hiemstra_LM has the highest MAP, indicating it performs slightly better in terms of precision across all queries.
2. **Reciprocal Rank**: Hiemstra_LM also has the highest reciprocal rank, suggesting it is more effective at retrieving the top relevant document.

3. **R-precision**: TF_IDF has the highest R-precision, indicating it performs better in terms of precision at the rank cut-off point.

Overall, Hiemstra_LM appears to be the most effective model based on MAP and Reciprocal Rank, while TF_IDF shows its strength in R-precision (at least in this case). BM25 performs consistently but slightly lower than the other two models in these metrics. This shows that different models have varying strengths in retrieval performance, and the choice of model can impact the effectiveness of the retrieval system.