Retrieval

September 27, 2023

1 Retrieval exercise

In this exercise, you will implement the query likelihood model with Jelinek-Mercer smoothing. This assignment builds on the previous assignment for creating a Pyserini index.

1.1 1. Build the index

Download the MS MARCO passage collection and build an index using Pyserini. This code is similar to PART 1 of the indexing assignment.

[ ]:
```
# NOTE: I comment out the pip install statements because I do them manually.
# NOTE: The file paths here may slightly alter from the original file paths.
    # This is so that the paths fit my local structure.

# !pip install pyserini
# !pip install faiss-cpu

!git clone https://github.com/castorini/anserini.git --recurse-submodules

# NOTE: Commented out because redownloading this file takes 20+ minutes.
# !wget https://msmarco.blob.core.windows.net/msmarcoranking/collection.tar.gz␣
 ↪-P Data/
!tar xvfz Data/collection.tar.gz -C Data/

!cd anserini && python tools/scripts/msmarco/convert_collection_to_jsonl.py \
 --collection-path ../Data/collection.tsv --output-folder ../Data/
 ↪collection_jsonl

!rm Data/*.tsv
!rm -rf sample_data

!python -m pyserini.index.lucene -collection JsonCollection -generator␣
 ↪DefaultLuceneDocumentGenerator -threads 9 \
-input Data/collection_jsonl -index Data/indexes/lucene-index-msmarco-passage␣
 ↪-storePositions -storeDocvectors -storeRaw
```

```
Cloning into 'anserini'…
remote: Enumerating objects: 29516, done.
```

```
remote: Counting objects: 100% (3834/3834), done.
remote: Compressing objects: 100% (1065/1065), done.
remote: Total 29516 (delta 3226), reused 3175 (delta 2681), pack-reused 25682
Receiving objects: 100% (29516/29516), 85.42 MiB | 10.14 MiB/s, done.
Resolving deltas: 100% (19875/19875), done.
Submodule 'tools' (https://github.com/castorini/anserini-tools.git) registered
for path 'tools'
Cloning into '/home/daanbrugmans/projects/ru-information-
retrieval-23-24/Assignments/anserini/tools'…
remote: Enumerating objects: 829, done.
remote: Counting objects: 100% (586/586), done.
remote: Compressing objects: 100% (498/498), done.
remote: Total 829 (delta 114), reused 549 (delta 87), pack-reused 243
Receiving objects: 100% (829/829), 192.26 MiB | 9.89 MiB/s, done.
Resolving deltas: 100% (198/198), done.
Submodule path 'tools': checked out '95d06f60043837a309331ffdbee7560dd1676313'
collection.tsv
Converting collection…
Converted 0 docs, writing into file 1
Converted 100,000 docs, writing into file 1
Converted 200,000 docs, writing into file 1
Converted 300,000 docs, writing into file 1
Converted 400,000 docs, writing into file 1
Converted 500,000 docs, writing into file 1
Converted 600,000 docs, writing into file 1
Converted 700,000 docs, writing into file 1
Converted 800,000 docs, writing into file 1
Converted 900,000 docs, writing into file 1
Converted 1,000,000 docs, writing into file 2
Converted 1,100,000 docs, writing into file 2
Converted 1,200,000 docs, writing into file 2
Converted 1,300,000 docs, writing into file 2
Converted 1,400,000 docs, writing into file 2
Converted 1,500,000 docs, writing into file 2
Converted 1,600,000 docs, writing into file 2
Converted 1,700,000 docs, writing into file 2
Converted 1,800,000 docs, writing into file 2
Converted 1,900,000 docs, writing into file 2
Converted 2,000,000 docs, writing into file 3
Converted 2,100,000 docs, writing into file 3
Converted 2,200,000 docs, writing into file 3
Converted 2,300,000 docs, writing into file 3
Converted 2,400,000 docs, writing into file 3
Converted 2,500,000 docs, writing into file 3
Converted 2,600,000 docs, writing into file 3
Converted 2,700,000 docs, writing into file 3
Converted 2,800,000 docs, writing into file 3
Converted 2,900,000 docs, writing into file 3
```

```
Converted 3,000,000 docs, writing into file 4
Converted 3,100,000 docs, writing into file 4
Converted 3,200,000 docs, writing into file 4
Converted 3,300,000 docs, writing into file 4
Converted 3,400,000 docs, writing into file 4
Converted 3,500,000 docs, writing into file 4
Converted 3,600,000 docs, writing into file 4
Converted 3,700,000 docs, writing into file 4
Converted 3,800,000 docs, writing into file 4
Converted 3,900,000 docs, writing into file 4
Converted 4,000,000 docs, writing into file 5
Converted 4,100,000 docs, writing into file 5
Converted 4,200,000 docs, writing into file 5
Converted 4,300,000 docs, writing into file 5
Converted 4,400,000 docs, writing into file 5
Converted 4,500,000 docs, writing into file 5
Converted 4,600,000 docs, writing into file 5
Converted 4,700,000 docs, writing into file 5
Converted 4,800,000 docs, writing into file 5
Converted 4,900,000 docs, writing into file 5
Converted 5,000,000 docs, writing into file 6
Converted 5,100,000 docs, writing into file 6
Converted 5,200,000 docs, writing into file 6
Converted 5,300,000 docs, writing into file 6
Converted 5,400,000 docs, writing into file 6
Converted 5,500,000 docs, writing into file 6
Converted 5,600,000 docs, writing into file 6
Converted 5,700,000 docs, writing into file 6
Converted 5,800,000 docs, writing into file 6
Converted 5,900,000 docs, writing into file 6
Converted 6,000,000 docs, writing into file 7
Converted 6,100,000 docs, writing into file 7
Converted 6,200,000 docs, writing into file 7
Converted 6,300,000 docs, writing into file 7
Converted 6,400,000 docs, writing into file 7
Converted 6,500,000 docs, writing into file 7
Converted 6,600,000 docs, writing into file 7
Converted 6,700,000 docs, writing into file 7
Converted 6,800,000 docs, writing into file 7
Converted 6,900,000 docs, writing into file 7
Converted 7,000,000 docs, writing into file 8
Converted 7,100,000 docs, writing into file 8
Converted 7,200,000 docs, writing into file 8
Converted 7,300,000 docs, writing into file 8
Converted 7,400,000 docs, writing into file 8
Converted 7,500,000 docs, writing into file 8
Converted 7,600,000 docs, writing into file 8
Converted 7,700,000 docs, writing into file 8
```

```
Converted 7,800,000 docs, writing into file 8
Converted 7,900,000 docs, writing into file 8
Converted 8,000,000 docs, writing into file 9
Converted 8,100,000 docs, writing into file 9
Converted 8,200,000 docs, writing into file 9
Converted 8,300,000 docs, writing into file 9
Converted 8,400,000 docs, writing into file 9
Converted 8,500,000 docs, writing into file 9
Converted 8,600,000 docs, writing into file 9
Converted 8,700,000 docs, writing into file 9
Converted 8,800,000 docs, writing into file 9
Done!
WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will
impact performance.
2023-09-27 19:56:44,599 INFO  [main] index.IndexCollection
(IndexCollection.java:380) - Setting log level to INFO
2023-09-27 19:56:44,603 INFO  [main] index.IndexCollection
(IndexCollection.java:383) - Starting indexer…
2023-09-27 19:56:44,603 INFO  [main] index.IndexCollection
(IndexCollection.java:384) - ============ Loading Parameters ============
2023-09-27 19:56:44,604 INFO  [main] index.IndexCollection
(IndexCollection.java:385) - DocumentCollection path: Data/collection_jsonl
2023-09-27 19:56:44,604 INFO  [main] index.IndexCollection
(IndexCollection.java:386) - CollectionClass: JsonCollection
2023-09-27 19:56:44,605 INFO  [main] index.IndexCollection
(IndexCollection.java:387) - Generator: DefaultLuceneDocumentGenerator
2023-09-27 19:56:44,605 INFO  [main] index.IndexCollection
(IndexCollection.java:388) - Threads: 9
2023-09-27 19:56:44,606 INFO  [main] index.IndexCollection
(IndexCollection.java:389) - Language: en
2023-09-27 19:56:44,606 INFO  [main] index.IndexCollection
(IndexCollection.java:390) - Stemmer: porter
2023-09-27 19:56:44,606 INFO  [main] index.IndexCollection
(IndexCollection.java:391) - Keep stopwords? false
2023-09-27 19:56:44,607 INFO  [main] index.IndexCollection
(IndexCollection.java:392) - Stopwords: null
2023-09-27 19:56:44,607 INFO  [main] index.IndexCollection
(IndexCollection.java:393) - Store positions? true
2023-09-27 19:56:44,608 INFO  [main] index.IndexCollection
(IndexCollection.java:394) - Store docvectors? true
2023-09-27 19:56:44,610 INFO  [main] index.IndexCollection
(IndexCollection.java:395) - Store document "contents" field? false
2023-09-27 19:56:44,611 INFO  [main] index.IndexCollection
(IndexCollection.java:396) - Store document "raw" field? true
2023-09-27 19:56:44,612 INFO  [main] index.IndexCollection
(IndexCollection.java:397) - Additional fields to index: []
2023-09-27 19:56:44,613 INFO  [main] index.IndexCollection
(IndexCollection.java:398) - Optimize (merge segments)? false
```

```
2023-09-27 19:56:44,614 INFO  [main] index.IndexCollection
(IndexCollection.java:399) - Whitelist: null
2023-09-27 19:56:44,615 INFO  [main] index.IndexCollection
(IndexCollection.java:400) - Pretokenized?: false
2023-09-27 19:56:44,616 INFO  [main] index.IndexCollection
(IndexCollection.java:401) - Index path: Data/indexes/lucene-index-msmarco-
passage
2023-09-27 19:56:44,623 INFO  [main] index.IndexCollection
(IndexCollection.java:481) - ============ Indexing Collection ============
2023-09-27 19:56:44,643 INFO  [main] index.IndexCollection
(IndexCollection.java:468) - Using DefaultEnglishAnalyzer
2023-09-27 19:56:44,644 INFO  [main] index.IndexCollection
(IndexCollection.java:469) - Stemmer: porter
2023-09-27 19:56:44,644 INFO  [main] index.IndexCollection
(IndexCollection.java:470) - Keep stopwords? false
2023-09-27 19:56:44,645 INFO  [main] index.IndexCollection
(IndexCollection.java:471) - Stopwords file: null
2023-09-27 19:56:44,836 INFO  [main] index.IndexCollection
(IndexCollection.java:510) - Thread pool with 9 threads initialized.
2023-09-27 19:56:44,836 INFO  [main] index.IndexCollection
(IndexCollection.java:512) - Initializing collection in Data/collection_jsonl
2023-09-27 19:56:44,845 INFO  [main] index.IndexCollection
(IndexCollection.java:521) - 9 files found
2023-09-27 19:56:44,846 INFO  [main] index.IndexCollection
(IndexCollection.java:522) - Starting to index…
2023-09-27 19:57:44,870 INFO  [main] index.IndexCollection
(IndexCollection.java:536) - 0.00% of files completed, 2,230,000 documents
indexed
2023-09-27 19:58:44,872 INFO  [main] index.IndexCollection
(IndexCollection.java:536) - 0.00% of files completed, 4,510,000 documents
indexed
2023-09-27 19:59:44,874 INFO  [main] index.IndexCollection
(IndexCollection.java:536) - 0.00% of files completed, 6,750,000 documents
indexed
2023-09-27 20:00:07,677 DEBUG [pool-2-thread-9]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
collection_jsonl/docs08.json: 841823 docs added.
2023-09-27 20:00:44,881 INFO  [main] index.IndexCollection
(IndexCollection.java:536) - 11.11% of files completed, 8,441,823 documents
indexed
2023-09-27 20:00:46,462 DEBUG [pool-2-thread-5]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
collection_jsonl/docs07.json: 1000000 docs added.
2023-09-27 20:00:51,110 DEBUG [pool-2-thread-7]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
collection_jsonl/docs01.json: 1000000 docs added.
2023-09-27 20:00:53,092 DEBUG [pool-2-thread-8]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
```

```
collection_jsonl/docs00.json: 1000000 docs added.
2023-09-27 20:00:56,970 DEBUG [pool-2-thread-6]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
collection_jsonl/docs04.json: 1000000 docs added.
2023-09-27 20:00:57,933 DEBUG [pool-2-thread-1]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
collection_jsonl/docs05.json: 1000000 docs added.
2023-09-27 20:00:58,058 DEBUG [pool-2-thread-2]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
collection_jsonl/docs02.json: 1000000 docs added.
2023-09-27 20:00:58,217 DEBUG [pool-2-thread-4]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
collection_jsonl/docs03.json: 1000000 docs added.
2023-09-27 20:01:05,544 DEBUG [pool-2-thread-3]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
collection_jsonl/docs06.json: 1000000 docs added.
2023-09-27 20:03:44,931 INFO  [main] index.IndexCollection
(IndexCollection.java:578) - Indexing Complete! 8,841,823 documents indexed
2023-09-27 20:03:44,932 INFO  [main] index.IndexCollection
(IndexCollection.java:579) - ============ Final Counter Values ============
2023-09-27 20:03:44,932 INFO  [main] index.IndexCollection
(IndexCollection.java:580) - indexed:        8,841,823
2023-09-27 20:03:44,933 INFO  [main] index.IndexCollection
(IndexCollection.java:581) - unindexable:            0
2023-09-27 20:03:44,934 INFO  [main] index.IndexCollection
(IndexCollection.java:582) - empty:                  0
2023-09-27 20:03:44,936 INFO  [main] index.IndexCollection
(IndexCollection.java:583) - skipped:                0
2023-09-27 20:03:44,937 INFO  [main] index.IndexCollection
(IndexCollection.java:584) - errors:                 0
2023-09-27 20:03:44,961 INFO  [main] index.IndexCollection
(IndexCollection.java:587) - Total 8,841,823 documents indexed in 00:07:00
```

1.2 2. Download and read the query file

You will rank MSMARCO passages for this set of queries.

[ ]:
```
!wget http://gem.cs.ru.nl/IR-Course/queries.txt -P Data/

queries = dict()
with open("Data/queries.txt", "r") as f:
    for line in f:
        cols = line.split("\t")
        queries[cols[0].strip()] = cols[1].strip()
```

```
--2023-09-27 20:03:45--  http://gem.cs.ru.nl/IR-Course/queries.txt
Resolving gem.cs.ru.nl (gem.cs.ru.nl)… 131.174.31.31
Connecting to gem.cs.ru.nl (gem.cs.ru.nl)|131.174.31.31|:80… connected.
```

```
HTTP request sent, awaiting response… 200 OK
Length: 2275 (2.2K) [text/plain]
Saving to: 'Data/queries.txt'

queries.txt         100%[==================>]   2.22K  --.-KB/s    in 0s

2023-09-27 20:03:46 (88.6 MB/s) - 'Data/queries.txt' saved [2275/2275]
```

1.3 3. Implement the retrieval model

You will implement language model with Jelinek-Mercer (JM) smoothing:

$$score(q,d) = \sum_{t \in q} log((1-\lambda)\frac{c(t,d)}{|d|} + \lambda\frac{c(t,C)}{|C|}),$$

where $c(t,d)$ and $c(t,C)$ represent frequency of a term in a document and collection, respectively.

Notes about your implementation: - Skip a term if it does not exist in the whole collection. This avoids $log(0)$. - Make sure to use the right form of a query (analyzed vs. not analyzed) - Use natural logarithm

1.3.1 3.1. Obtain collection length

In this code, the global variable `len_C` denotes collection length.

[ ]:
```python
from pyserini.index.lucene import IndexReader

global len_C

# =======Your code=======
index_reader = IndexReader("Data/indexes/lucene-index-msmarco-passage")
len_C = index_reader.stats()["total_terms"]
# ========================
```

Run this to test your code. If everything is correct, you should not get errors here.

[ ]:
```python
assert len_C == 352316036
```

1.3.2 3.2. Obtain document length

Here you need compute the length of document (as it is stored in the index).

Hint: You first need to get the document vector from your Pyserini index. Consult Pyseniri documentation to find the right function.

[ ]:
```python
def len_doc(d):
    # =======Your code=======
    document_vector = index_reader.get_document_vector(d)
    len_d = sum([value for _, value in document_vector.items()])
    # ========================
```

```
    return len_d
```

[ ]:
```
# Test your code
assert len_doc("2674124") == 31
```

1.3.3 3.3. Obtain collection frequency of a term

Obtain number of times a term appers in the whole collection.

[ ]:
```
def coll_freq(t):
    # =======Your code=======
    cf = index_reader.get_term_counts(t)[1]
    # ======================
    return cf
```

[ ]:
```
# Test your code
assert coll_freq("record") == 226439
```

1.3.4 3.4. Obtain term frequency

Obtain number of times a term appears in a document.

[ ]:
```
def term_freq(t, d):
    # =======Your code=======
    document_vector = index_reader.get_document_vector(d)

    if t not in document_vector:
        return 0

    tf = document_vector[t]
    # ======================
    return tf
```

[ ]:
```
# Test your code
assert term_freq("record", "2674124") == 2
assert term_freq("presence", "2674124") == 0
```

1.3.5 3.5. Compute JM-smoothed probability for a single term

Here, you need to implement the following formula:

$$P_{JM}(t,d) = (1-\lambda)\frac{c(t,d)}{|d|} + \lambda\frac{c(t,C)}{|C|}$$

[ ]:
```
def prob_t_Md(t, d, lambd):
    # =======Your code=======
```

```
    p_t_Md = (1 - lambd) * term_freq(t, d)/len_doc(d) + lambd * coll_freq(t)/
↪len_C
    # ========================
    return p_t_Md
```

[ ]:
```
# Test your code
assert prob_t_Md("record", "2674124", 0.1) == 0.05812878768549357
assert prob_t_Md("darcig", "2674124", 0.1) == 0
```

1.3.6 3.6. Compute JM-smoothed probability for a query

[ ]:
```
import math

def score_doc(q, d, lambd):
    # =======Your code=======
    p_q_Md = 0

    for term in q:
        prob_t_Md_for_term = prob_t_Md(term, d, lambd)

        if prob_t_Md_for_term != 0:
            p_q_Md += math.log(prob_t_Md_for_term)
    # ========================
    return p_q_Md
```

[ ]:
```
q1 = index_reader.analyze("are naturalization records public")
q2 = index_reader.analyze("kemeet land")
doc = "2674124"
assert score_doc(q1, doc, 0.1) == -9.227787624348021
assert score_doc(q2, doc, 0.1) == -10.254756777887694
```

1.4 4. Rank documents for the given queries

Ranking is done in two steps: 1. First pass retrieval: Use a fast ranker (i.e., Pyserini LuceneSearcher) ro rank all documents for a given query. 2. Second pass retrieval: Re-rank top-100 documents from the 1st pass retrieval using your retrieval model. This is to make the ranking process efficient.

Notes: - You need to change the default values of LuceneSearcher functions to obtain top-100 documents - Set the value of lambda to 0.1 - Store your final ranking results in the `results` variable. Every item in the `results` list is a list containing queryID, documentID, and score. This is an example how the content of results should look like:

```
[['23849', '4348282', -10.65], ['23849', '7119957', -12.63], ['23849', '',
-17.687729001682484], ...]
```

[ ]:
```python
from pyserini.search.lucene import LuceneSearcher

results = []
searcher = LuceneSearcher("Data/indexes/lucene-index-msmarco-passage")
for qid, q in queries.items():
    # =======Your code=======
    lucene_searcher_top_100_hits = searcher.search(q, 100)
    analyzed_query = index_reader.analyze(q)

    for lucene_searcher_hit in lucene_searcher_top_100_hits:
        score = score_doc(analyzed_query, lucene_searcher_hit.docid, 0.1)
        results.append([qid, lucene_searcher_hit.docid, score])
    # ======================
```

```
/home/daanbrugmans/projects/ru-information-
retrieval-23-24/env/lib/python3.10/site-packages/tqdm/auto.py:21: TqdmWarning:
IProgress not found. Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

[ ]:
```python
# Test your code
assert round(sum([item[2] for item in results]), 3) == -160109.875
```

Write your results into a file. Submit this file together with the completed notebook.

[ ]:
```python
# check duplicates
check = set()
for res in results:
    if ((res[0], res[1])) in check:
        raise Exception("Error: Duplicate query-doc is found", res[0], res[1])
    check.add((res[0], res[1]))

# write results in a file
output_str = "\n".join([l[0] + "\tQ0\t" + l[1] + "\t0\t" + str(l[2]) +␣
 ↪"\tlm_jm" for l in results])
open("lm_jm.run", "w").write(output_str)
```

[ ]:
```
246890
```

1.5 Handing in

Submit the result file (ranked documents), the filled-in notebook, and the pdf version of your notebook:

- The result file should be named STUDENTNUMBER_FIRSTNAME_LASTNAME_lm_jm.run
- The notebook should be named STUDENTNUMBER_FIRSTNAME_LASTNAME_retrieval.ipynb
- The pdf version of your notebook should be named STUDENTNUMBER_FIRSTNAME_LASTNAME_retrieval.pdf

[ ]: