Indexing

September 20, 2023

1 Indexing Excercise

This exercise has two parts:

- In part 1, we are going to index the MS MARCO passage collection Pyserini toolkit and explore some features of the index. For this part, you only need to run code and understand it. You will be using the index and code snippets in the next assignment.
- In part 2, we are going to write a code for generating an inverted index and index part of MS MARCO collection. For this part, you need to first run the first part (1.1 and 1.2) to build the environment and prepare the data.

1.1 PART 1: Generate the index via Pyserini

We use Anserini toolkit and its python interface Pyserini to run our experiments.

***This part is created based on Anserini/Pyserini tutorials. You can learn more by checking their repositories and tutorials.*

1.1.1 1.1 Setup the environment

Install Pyserini:

[ ]:
```
# !pip install pyserini
# NOTE: I commented the pip install statement out
    # because I run the pip installs manually in the terminal
# NOTE: I altered some paths for saving and loading indexes and data
    # so that it fits my folder structure.
    # Functionality should be the exact same.
```

Clone the Anserini repository from GitHub:

[1]:
```
!git clone https://github.com/castorini/anserini.git
!cd anserini && git checkout ad5ba1c76196436f8a0e28efdb69960d4873efe3
```

```
fatal: destination path 'anserini' already exists and is not an empty directory.
HEAD is now at ad5ba1c7 Release notes for v0.9.2 (#1197)
```

1.1.2 1.2 Get the collection and prepare the files

MS MARCO (MicroSoft MAchine Reading COmprehension) is a large-scale dataset that defines many tasks from question answering to ranking. Here we focus on the collection designed for passage re-ranking.

[ ]:
```
!wget https://msmarco.blob.core.windows.net/msmarcoranking/collection.tar.gz -P␣
 ↪Data/
```

[2]:
```
!ls Data/
!tar xvfz Data/collection.tar.gz -C Data/
```

```
CollectionJsonl  collection.tar.gz  indexes
collection.tsv
```

The original MS MARCO collection is a tab-separated values (TSV) file. We need to convert the collection into the jsonl format that can be processed by Anserini. jsonl files contain JSON object per line.

This command generates 9 jsonl files in your data/msmarco_passage/collection_jsonl directory, each with 1M lines (except for the last one, which should have 841,823 lines).

[3]:
```
!cd anserini && python ./src/main/python/msmarco/convert_collection_to_jsonl.py␣
 ↪\
 --collection_path ../Data/collection.tsv --output_folder ../Data/
 ↪CollectionJsonl
```

```
Converting collection…
Converted 0 docs in 1 files
Converted 100000 docs in 1 files
Converted 200000 docs in 1 files
Converted 300000 docs in 1 files
Converted 400000 docs in 1 files
Converted 500000 docs in 1 files
Converted 600000 docs in 1 files
Converted 700000 docs in 1 files
Converted 800000 docs in 1 files
Converted 900000 docs in 1 files
Converted 1000000 docs in 2 files
Converted 1100000 docs in 2 files
Converted 1200000 docs in 2 files
Converted 1300000 docs in 2 files
Converted 1400000 docs in 2 files
Converted 1500000 docs in 2 files
Converted 1600000 docs in 2 files
Converted 1700000 docs in 2 files
Converted 1800000 docs in 2 files
Converted 1900000 docs in 2 files
Converted 2000000 docs in 3 files
Converted 2100000 docs in 3 files
Converted 2200000 docs in 3 files
Converted 2300000 docs in 3 files
Converted 2400000 docs in 3 files
Converted 2500000 docs in 3 files
Converted 2600000 docs in 3 files
Converted 2700000 docs in 3 files
Converted 2800000 docs in 3 files
Converted 2900000 docs in 3 files
Converted 3000000 docs in 4 files
Converted 3100000 docs in 4 files
Converted 3200000 docs in 4 files
Converted 3300000 docs in 4 files
Converted 3400000 docs in 4 files
Converted 3500000 docs in 4 files
Converted 3600000 docs in 4 files
Converted 3700000 docs in 4 files
Converted 3800000 docs in 4 files
Converted 3900000 docs in 4 files
Converted 4000000 docs in 5 files
Converted 4100000 docs in 5 files
Converted 4200000 docs in 5 files
Converted 4300000 docs in 5 files
Converted 4400000 docs in 5 files
Converted 4500000 docs in 5 files
Converted 4600000 docs in 5 files
Converted 4700000 docs in 5 files
Converted 4800000 docs in 5 files
Converted 4900000 docs in 5 files
Converted 5000000 docs in 6 files
Converted 5100000 docs in 6 files
Converted 5200000 docs in 6 files
Converted 5300000 docs in 6 files
Converted 5400000 docs in 6 files
Converted 5500000 docs in 6 files
Converted 5600000 docs in 6 files
Converted 5700000 docs in 6 files
Converted 5800000 docs in 6 files
Converted 5900000 docs in 6 files
Converted 6000000 docs in 7 files
Converted 6100000 docs in 7 files
Converted 6200000 docs in 7 files
Converted 6300000 docs in 7 files
Converted 6400000 docs in 7 files
Converted 6500000 docs in 7 files
Converted 6600000 docs in 7 files
Converted 6700000 docs in 7 files
Converted 6800000 docs in 7 files
Converted 6900000 docs in 7 files
Converted 7000000 docs in 8 files
Converted 7100000 docs in 8 files
Converted 7200000 docs in 8 files
Converted 7300000 docs in 8 files
Converted 7400000 docs in 8 files
Converted 7500000 docs in 8 files
Converted 7600000 docs in 8 files
Converted 7700000 docs in 8 files
Converted 7800000 docs in 8 files
Converted 7900000 docs in 8 files
Converted 8000000 docs in 9 files
Converted 8100000 docs in 9 files
Converted 8200000 docs in 9 files
Converted 8300000 docs in 9 files
Converted 8400000 docs in 9 files
Converted 8500000 docs in 9 files
Converted 8600000 docs in 9 files
Converted 8700000 docs in 9 files
Converted 8800000 docs in 9 files
Done!
```

Check the data!

jsonl files are JSON files with keys id and contents:

```
[4]: !wc -l Data/CollectionJsonl/* -l Data/CollectionJsonl/*
```

```
   1000000 Data/CollectionJsonl/docs00.json
   1000000 Data/CollectionJsonl/docs01.json
   1000000 Data/CollectionJsonl/docs02.json
   1000000 Data/CollectionJsonl/docs03.json
   1000000 Data/CollectionJsonl/docs04.json
   1000000 Data/CollectionJsonl/docs05.json
   1000000 Data/CollectionJsonl/docs06.json
   1000000 Data/CollectionJsonl/docs07.json
    841823 Data/CollectionJsonl/docs08.json
         2 Data/CollectionJsonl/text_index.txt
    698784 Data/CollectionJsonl/tiny_index.txt
   1000000 Data/CollectionJsonl/docs00.json
   1000000 Data/CollectionJsonl/docs01.json
   1000000 Data/CollectionJsonl/docs02.json
   1000000 Data/CollectionJsonl/docs03.json
   1000000 Data/CollectionJsonl/docs04.json
   1000000 Data/CollectionJsonl/docs05.json
   1000000 Data/CollectionJsonl/docs06.json
   1000000 Data/CollectionJsonl/docs07.json
    841823 Data/CollectionJsonl/docs08.json
         2 Data/CollectionJsonl/text_index.txt
   698784 Data/CollectionJsonl/tiny_index.txt
  19081218 total
```

[5]:
```
!head -5 Data/CollectionJsonl/docs00.json
```

```
{"id": "0", "contents": "The presence of communication amid scientific minds was
equally important to the success of the Manhattan Project as scientific
intellect was. The only cloud hanging over the impressive achievement of the
atomic researchers and engineers is what their success truly meant; hundreds of
thousands of innocent lives obliterated."}
{"id": "1", "contents": "The Manhattan Project and its atomic bomb helped bring
an end to World War II. Its legacy of peaceful uses of atomic energy continues
to have an impact on history and science."}
{"id": "2", "contents": "Essay on The Manhattan Project - The Manhattan Project
The Manhattan Project was to see if making an atomic bomb possible. The success
of this project would forever change the world forever making it known that
something this powerful can be manmade."}
{"id": "3", "contents": "The Manhattan Project was the name for a project
conducted during World War II, to develop the first atomic bomb. It refers
specifically to the period of the project from 194 \u00e2\u0080\u00a6 2-1946
under the control of the U.S. Army Corps of Engineers, under the administration
of General Leslie R. Groves."}
{"id": "4", "contents": "versions of each volume as well as complementary
websites. The first website\u00e2\u0080\u0093The Manhattan Project: An
Interactive History\u00e2\u0080\u0093is available on the Office of History and
Heritage Resources website, http://www.cfo. doe.gov/me70/history. The Office of
History and Heritage Resources and the National Nuclear Security"}
```

Remove the original files to make room for the index. Check the contents of `data/msmarco_passage` before and after.

[6]:
```
!ls Data
!rm Data/*.tsv
!ls Data
!rm -rf sample_data
```

```
CollectionJsonl  collection.tar.gz  collection.tsv  indexes
CollectionJsonl  collection.tar.gz  indexes
```

1.1.3 1.3 Generate the index using Pyserini

Here are some common indexing options with Pyserini (for more options, check Pyserini documentation):

```
* input: Path to collection
* threads: Number of threads to run
* collection: Type of Anserini Collection, e.g., LuceneDocumentGenerator, TweetGenerator (subcl
* index: Path to index output
* storePositions: Boolean flag to store positions
* storeDocvectors: Boolean flag to store document vectors
* storeRawDocs: Boolean flag to store raw document text
* keepStopwords: Boolean flag to keep stopwords (False by default)
* stemmer: Stemmer to use (Porter by default)
```

We now have everything in place to index the collection. **The indexing speed may vary, the process may take about 10 minutes (or more) in Google Colab.**

[7]:
```
!python3 -m pyserini.index -collection JsonCollection -generator␣
 ↪DefaultLuceneDocumentGenerator -threads 9 \
-input Data/CollectionJsonl -index Data/indexes/lucene-index-msmarco-passage␣
 ↪-storePositions -storeDocvectors -storeRaw
```

```
pyserini.index is deprecated, please use pyserini.index.lucene.
WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will
impact performance.
2023-09-20 17:07:41,153 INFO  [main] index.IndexCollection
(IndexCollection.java:380) - Setting log level to INFO
2023-09-20 17:07:41,157 INFO  [main] index.IndexCollection
(IndexCollection.java:383) - Starting indexer…
2023-09-20 17:07:41,157 INFO  [main] index.IndexCollection
(IndexCollection.java:384) - ============ Loading Parameters ============
2023-09-20 17:07:41,157 INFO  [main] index.IndexCollection
(IndexCollection.java:385) - DocumentCollection path: Data/CollectionJsonl
2023-09-20 17:07:41,158 INFO  [main] index.IndexCollection
(IndexCollection.java:386) - CollectionClass: JsonCollection
2023-09-20 17:07:41,158 INFO  [main] index.IndexCollection
(IndexCollection.java:387) - Generator: DefaultLuceneDocumentGenerator
2023-09-20 17:07:41,158 INFO  [main] index.IndexCollection
(IndexCollection.java:388) - Threads: 9
2023-09-20 17:07:41,159 INFO  [main] index.IndexCollection
(IndexCollection.java:389) - Language: en
2023-09-20 17:07:41,159 INFO  [main] index.IndexCollection
(IndexCollection.java:390) - Stemmer: porter
2023-09-20 17:07:41,159 INFO  [main] index.IndexCollection
(IndexCollection.java:391) - Keep stopwords? false
2023-09-20 17:07:41,160 INFO  [main] index.IndexCollection
(IndexCollection.java:392) - Stopwords: null
2023-09-20 17:07:41,160 INFO  [main] index.IndexCollection
(IndexCollection.java:393) - Store positions? true
2023-09-20 17:07:41,160 INFO  [main] index.IndexCollection
(IndexCollection.java:394) - Store docvectors? true
2023-09-20 17:07:41,161 INFO  [main] index.IndexCollection
(IndexCollection.java:395) - Store document "contents" field? false
2023-09-20 17:07:41,161 INFO  [main] index.IndexCollection
(IndexCollection.java:396) - Store document "raw" field? true
2023-09-20 17:07:41,161 INFO  [main] index.IndexCollection
(IndexCollection.java:397) - Additional fields to index: []
2023-09-20 17:07:41,162 INFO  [main] index.IndexCollection
```

```
(IndexCollection.java:398) - Optimize (merge segments)? false
2023-09-20 17:07:41,162 INFO  [main] index.IndexCollection
(IndexCollection.java:399) - Whitelist: null
2023-09-20 17:07:41,163 INFO  [main] index.IndexCollection
(IndexCollection.java:400) - Pretokenized?: false
2023-09-20 17:07:41,163 INFO  [main] index.IndexCollection
(IndexCollection.java:401) - Index path: Data/indexes/lucene-index-msmarco-
passage
2023-09-20 17:07:41,166 INFO  [main] index.IndexCollection
(IndexCollection.java:481) - ============ Indexing Collection ============
2023-09-20 17:07:41,184 INFO  [main] index.IndexCollection
(IndexCollection.java:468) - Using DefaultEnglishAnalyzer
2023-09-20 17:07:41,184 INFO  [main] index.IndexCollection
(IndexCollection.java:469) - Stemmer: porter
2023-09-20 17:07:41,185 INFO  [main] index.IndexCollection
(IndexCollection.java:470) - Keep stopwords? false
2023-09-20 17:07:41,185 INFO  [main] index.IndexCollection
(IndexCollection.java:471) - Stopwords file: null
2023-09-20 17:07:41,381 INFO  [main] index.IndexCollection
(IndexCollection.java:510) - Thread pool with 9 threads initialized.
2023-09-20 17:07:41,381 INFO  [main] index.IndexCollection
(IndexCollection.java:512) - Initializing collection in Data/CollectionJsonl
2023-09-20 17:07:41,384 INFO  [main] index.IndexCollection
(IndexCollection.java:521) - 9 files found
2023-09-20 17:07:41,385 INFO  [main] index.IndexCollection
(IndexCollection.java:522) - Starting to index…
2023-09-20 17:08:41,596 INFO  [main] index.IndexCollection
(IndexCollection.java:536) - 0.00% of files completed, 2,970,000 documents
indexed
2023-09-20 17:09:41,604 INFO  [main] index.IndexCollection
(IndexCollection.java:536) - 0.00% of files completed, 6,080,000 documents
indexed
2023-09-20 17:10:03,787 DEBUG [pool-2-thread-9]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
CollectionJsonl/docs08.json: 841823 docs added.
2023-09-20 17:10:35,870 DEBUG [pool-2-thread-5]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
CollectionJsonl/docs07.json: 1000000 docs added.
2023-09-20 17:10:41,615 INFO  [main] index.IndexCollection
(IndexCollection.java:536) - 22.22% of files completed, 8,601,823 documents
indexed
2023-09-20 17:10:44,567 DEBUG [pool-2-thread-6]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
CollectionJsonl/docs04.json: 1000000 docs added.
2023-09-20 17:10:45,201 DEBUG [pool-2-thread-2]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
CollectionJsonl/docs02.json: 1000000 docs added.
2023-09-20 17:10:45,288 DEBUG [pool-2-thread-4]
```

```
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
CollectionJsonl/docs03.json: 1000000 docs added.
2023-09-20 17:10:45,639 DEBUG [pool-2-thread-7]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
CollectionJsonl/docs01.json: 1000000 docs added.
2023-09-20 17:10:45,683 DEBUG [pool-2-thread-1]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
CollectionJsonl/docs05.json: 1000000 docs added.
2023-09-20 17:10:46,091 DEBUG [pool-2-thread-8]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
CollectionJsonl/docs00.json: 1000000 docs added.
2023-09-20 17:10:47,225 DEBUG [pool-2-thread-3]
index.IndexCollection$LocalIndexerThread (IndexCollection.java:345) -
CollectionJsonl/docs06.json: 1000000 docs added.
2023-09-20 17:12:10,552 INFO  [main] index.IndexCollection
(IndexCollection.java:578) - Indexing Complete! 8,841,823 documents indexed
2023-09-20 17:12:10,553 INFO  [main] index.IndexCollection
(IndexCollection.java:579) - ============ Final Counter Values ============
2023-09-20 17:12:10,553 INFO  [main] index.IndexCollection
(IndexCollection.java:580) - indexed:         8,841,823
2023-09-20 17:12:10,553 INFO  [main] index.IndexCollection
(IndexCollection.java:581) - unindexable:             0
2023-09-20 17:12:10,553 INFO  [main] index.IndexCollection
(IndexCollection.java:582) - empty:                   0
2023-09-20 17:12:10,554 INFO  [main] index.IndexCollection
(IndexCollection.java:583) - skipped:                 0
2023-09-20 17:12:10,554 INFO  [main] index.IndexCollection
(IndexCollection.java:584) - errors:                  0
2023-09-20 17:12:10,560 INFO  [main] index.IndexCollection
(IndexCollection.java:587) - Total 8,841,823 documents indexed in 00:04:29
```

Check the size of the index at the specified destination:

[8]:
```
!ls Data/indexes
!du -h Data/indexes/lucene-index-msmarco-passage
```

```
lucene-index-msmarco-passage
4.3G    Data/indexes/lucene-index-msmarco-passage
```

1.1.4 1.4 Explore Pyserini index

We can now explore the index using the The IndexReader class of Pyserini.

Read Usage of the Index Reader API notebook for more information on accessing and manipulating an inverted index.

[9]:
```python
from pyserini.index import IndexReader

index_reader = IndexReader('Data/indexes/lucene-index-msmarco-passage')
```

Compute the collection and document frequencies of a term:

[10]:
```
term = 'played'

# Look up its document frequency (df) and collection frequency (cf).
# Note, we use the unanalyzed form:
df, cf = index_reader.get_term_counts(term)

analyzed_form = index_reader.analyze(term)
print(f'Analyzed form of term "{analyzed_form[0]}": df={df}, cf={cf}')
```

```
Analyzed form of term "plai": df=155044, cf=200696
```

Get basic index statistics of the index.

Note that unless the underlying index was built with the `-optimize` option (i.e., merging all index segments into a single segment), unique_terms will show -1 (think what could be reason).

[11]:
```
index_reader.stats()
```

[11]:
```
{'total_terms': 352316036,
 'documents': 8841823,
 'non_empty_documents': 8841823,
 'unique_terms': -1}
```

1.2 PART 2: Generate the index yourself

1.2.1 2.1 Processing the text

We need to process the text, which includes tokenization, stopword removal, and lowercasing.

[12]:
```
STOPWORDS = ['a', 'an', 'and', 'are', 'as', 'at', 'be', 'but', 'by', 'for',␣
 ↪'if', 'in', 'into', 'is', 'it', 'no', 'not', 'of', 'on', 'or', 'such',␣
 ↪'that', 'the', 'their', 'then', 'there', 'these', 'they', 'this', 'to',␣
 ↪'was', 'will', 'with']

def process(text):
    terms = []
    # Remove special characters
    chars = ['\'', '.', ':', ',', '!', '?', '(', ')']
    for ch in chars:
        if ch in text:
            text = text.replace(ch, ' ')

    # Lowercasing and stopword removal
    for term in text.split():
        term = term.lower()
        if term not in STOPWORDS:
            terms.append(term)
    return terms
```

1.2.2 2.2 Complete the code for Inverted Index

Implement the InvertedIndex class.

Write the index to a file, where posting list of each term is presented in a line with this format: `Term1 docID1:freq1 docID2:freq2 ...`, e.g.,

```
term1 1:1 4:2 5:1
term2 2:1
term3 1:3 3:3 9:2
...
```

[13]:
```
import os

class InvertedIndex(object):
    def __init__(self):
        self.index = {}

    def add_posting(self, term:str, doc_id:int, count:int):
        """Adds a posting (term and Document ID) to the index."""
        # =======Your code=======
        if self.get_posting(term) is None:
            self.index[term] = list()

        self.index[term].append([doc_id, count])
        # =======================

    def get_posting(self,term:str):
        """Returns the posting list of the term from the index."""
        # =======Your code=======
        return self.index.get(term)
        # =======================

    def get_dictionary(self):
        """Returns the dictionary of the index (unique terms in the index)."""
        # =======Your code=======
        return self.index.keys()
        # =======================

    def write_to_file(self, filename_index:str):
        """Writes the index to a textfile."""
        # =======Your code=======
        path_to_index = os.path.join(os.getcwd(), filename_index)
        index_file = open(path_to_index, "w")
        alphabetically_sorted_index = dict(sorted(self.index.items()))
        for key, value in alphabetically_sorted_index.items():
            index_file.write(f"{key}")

            for doc_id, count in value:
                index_file.write(f" {doc_id}:{count}")

            index_file.write("\n")

        index_file.close()
        # =======================
```

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

```
[14]: index = InvertedIndex()
      index.add_posting("t1", 1, 2)
      index.add_posting("t1", 2, 1)
      index.add_posting("t2", 2, 3)
      assert len(index.get_dictionary()) == 2
      assert len(index.get_posting("t1")) == 2
      assert index.get_posting("t3") == None
      index.write_to_file("Data/CollectionJsonl/text_index.txt")
```

1.2.3 2.3 Index part of the MS MARCO collection

Complete the code to process the text and create the index. Note that we are only interested in indexing `docs00.json` file and it takes few minutes to create the index.

```
[15]: import collections
      import json

      ind = InvertedIndex()
      file = "Data/CollectionJsonl/docs00.json"
      index_file = "Data/CollectionJsonl/tiny_index.txt"

      def index(jsonl_file):
          with open(jsonl_file, 'r') as f:
              for line in f:
                  doc = json.loads(line)
                  # =======Your code=======
                  document_id, document_text = doc.values()
                  document_id = int(document_id)

                  cleaned_document_text = process(document_text)
                  cleaned_document_text_without_duplicate_terms =␣
       ↪set(cleaned_document_text)
```

```
            for term in cleaned_document_text_without_duplicate_terms:
                term_count = cleaned_document_text.count(term)
                ind.add_posting(term, document_id, term_count)
            # =======================

index(file)
ind.write_to_file(index_file)
```

Run this to test your code.

[16]:
```
with open(index_file, 'r') as fp:
    assert len(fp.readlines()) == 698784

assert len(ind.get_posting("pressingly")) == 3
assert len(ind.get_posting("veada")) == 2
```

1.3 Handing in

Hand in both the result file and the filled-in notebook:

- The result file should be named STUDENTNUMBER_FIRSTNAME_LASTNAME_tiny_index.txt
- The notebook should be named STUDENTNUMBER_FIRSTNAME_LASTNAME_indexing.ipynb

[ ]:
```

```