Assignment 4: Inverted Index

Special Topics in Computer Science: Big Data Systems
CS-UH 3260 Spring 2023
MAX 10 points

An inverted index is a data structure used to facilitate quick and efficient search in large document collections. In this assignment, you will be implementing a MapReduce job to create such a data structure for a collection of Wikipedia articles, as well as a basic search interface.

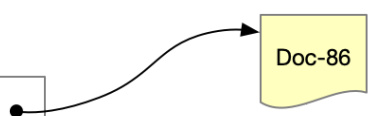
1. System installation (not graded):

- Install the Python package MRJob (and any required dependency). This library is a good simulation of MapReduce framework that runs locally. The same script you produce can be used on an actual Hadoop cluster or on the cloud.
    - **pip install mrjob**
- Once your script is ready you can run it and generate the output using a similar command:
    - **python YourIndexBuilder.py ./documents –output ./out_dir**
- For further documentation: https://mrjob.readthedocs.io/en/latest/

2. Description

An inverted index is a data structure used to efficiently store and retrieve information about terms and their occurrences in a collection of documents. It maps each unique term to a list of documents (or positions within documents) where the term appears. Inverted indexes are fundamental in information retrieval systems, such as search engines, enabling fast and accurate text-based searches. In practice, it can be implemented as a dictionary of dictionaries:

hello	Doc-1, 100	Doc-8, 8	Doc-1001, 1			
world	Doc-11, 20	Doc-8, 2	Doc-18, 2	Doc-86, 1		
new york	Doc-23, 3	Doc-233, 2	Doc-1, 7	Doc-2, 1		
paris	Doc-49, 1	Doc-11, 2	Doc-5, 2	Doc-77, 1	Doc-86, 2	Doc-22, 2
tokyo	Doc-13, 1	Doc-22, 2	Doc-5, 2	Doc-77, 5	Doc-32, 2	Doc-1001, 1
dubai	Doc-1001, 2					



Requirements:

1. Create an inverted index with map-reduce and compute relevance data, and other information needed for your system.
2. Load the index in memory using a suitable data structure or system, focusing on a specific vocabulary set.
3. Develop a prompt interface (e.g., terminal such as the one below) for **"single keyword"** input, searching the inverted index and returning the top-10 relevant documents.
4. Use provided test data for code development, adjusting the file count as needed.

```
> Enter Search Term: "world"
Doc-11 (Article: Economy) #.. digital world continues to transform ..
Doc-8  (Article: Culture)  #.. cultures around the world contribute ..
Doc-18 (Article: Future) #.. affect the entire world, necessitating..
Doc-86 (Article: Technology) #.. the world of technology offers ..
About 4 documents (in 0.71 seconds)
> Enter Search Term: _
```

Note on Relevance, The TFIDF score:
The widely-used relevance score in information retrieval, *term-frequency inverse-document frequency* (TFIDF), is determined by multiplying a keyword's frequency (TF) by its inverse document frequency (IDF) across a document collection. This value indicates a term's significance to a document in a collection, with common terms like "*with*" being less important due to their prevalence in multiple documents.

Data and vocabulary:

- You will be working with Wikipedia articles. 10,000 articles were collected and organized for you. Google Drive
- It is recommended that you use a tokenizer: https://www.nltk.org/api/nltk.tokenize.html
- Consider adding the following functionalities:
  - Fixed vocabulary i.e., only words present in a dictionary.
  - Word stemming

```python
import nltk

# Loads english words
from nltk.corpus import words
words = set(stopwords.words('english'))

# Sentence tokenizer
from nltk.tokenize import word_tokenize

# word stemming
from nltk.stem import SnowballStemmer
stemmer = SnowballStemmer("english")
print(stemmer.stem('iteration'))
# prints: iter
```

🚨 You cannot use an existing implementation or copy code.🚨
😎Have fun coding the fundamental component of a document search engine! 😎

3 Deliverables and Grading

🎯 **Deliverable:** Python Code + Report with any special instructions

Description	**Score (/10)**
• Correct implementation of inverted index creation using MapReduce (ie., map and reduce functions to tokenize documents and generate the inverted index.) • Efficient loading of the inverted index into memory using the appropriate data structures and/or system (e.g., Memcached). • Creation of a simple interface (terminal) for keyword search. ○ Alternatively, a Web app (e.g., using Flask) • Proper implementation of search functionality on the inverted index, ie., **top 10** relevant documents based on the given keyword(s). • Accurate ordering of search results by relevance TFIDF	7
• Proper error handling	1
• The code works seamlessly on test data and large data.	1
• Documentation: clear code comments + 1 short report explaining the implementation and design choices	1

• Bonus points 🙌: with a bit more effort you could:
  ○ Support **multiple word search** by computing result intersection and leveraging the TFIDF score.
  ○ Show a snippet of text around the first occurrence of the word in the document. e.g., ±3 words before and after the search query.
	2

** Bonus point is used to pad your score up to the maximum score, but the total score of this assignment cannot exceed 10 points.*