Report

Instructions to run the code:

- python3 final.py documents10k
- Default Memcached Server and Port: localhost 11212 (can be modified in the global variables section)

The code implements a MapReduce job to create an inverted index for a corpus of text documents. It uses the MRJob library and the Natural Language Toolkit (NLTK) to perform text preprocessing tasks such as tokenization, stemming, and stop word removal. It also uses the pymemcache library to store the inverted index in memory and provide a search interface for the user.

The code starts by importing necessary libraries such as MRJob, RegexpTokenizer, SnowballStemmer, stopwords, os, json, sys, and pymemcache. It also defines some global variables such as the number of documents in the corpus, IP, and PORT. It creates a dictionary of valid English words, converts them to lowercase for the ease of comparison, and removes stopwords.

The code then creates a connection to the Memcached server using the IP and PORT provided in the global variables. It also flushes all the key-value pairs in the cache to ensure that there are no old values present.

The InvertedIndex class is then defined, which extends the MRJob class. The constructor initializes a SnowballStemmer for stemming the words and an empty dictionary for storing the inverted index.
The mapper function takes a line of text as input, which is a single line from a document, and emits key-value pairs where the key is the stemmed word and the value is the document ID and frequency. It uses the NLTK RegexpTokenizer to tokenize the text into words, removes stopwords and checks if the word is a valid word in the English dictionary. If the word is a valid word, it is stemmed using the SnowballStemmer and emitted as a key-value pair.

The reducer function takes a key-value pair where the key is a stemmed word, and the value is a list of document IDs and their frequencies. It first counts the total number of documents that contain the word and concatenates the document IDs and their frequencies into a dictionary. It then calculates the inverse document frequency (idf) for the word and multiplies the word frequency by idf to get the tf-idf score. The dictionary of document IDs and their tf-idf scores is sorted based on decreasing tf-idf scores, and the resulting dictionary is stored in the inverted index dictionary.

The reducer_final function is called after all the reducer functions are executed. Rather than writing the inverted index to Memcached in the reducer function, in order to improve performance, the code first stores the inverted index in a dictionary in memory within the reducer function, and then write the entire inverted index to Memcached at the end of the job using the set_multi function.
The main function checks if the correct number of arguments is provided when running the script. It takes a directory / file name as an argument. It then runs the InvertedIndex job by calling the run function.

After the InvertedIndex job is executed, the main function enters a loop where it prompts the user to enter a keyword to be searched. If the input is not a single word, the user is prompted to enter a single word. If the keyword is "exit0", the loop is broken, and the program terminates. If the keyword is a valid word, the program searches the inverted index for the keyword and displays up to top 10 documents based on their tf-idf scores.

The execution time is around 1.5 hours for 10K files.