Homework #1 - MapReduce

Due: 4/3/23

Content Covered
MapReduce

Homework Overview

For this homework you will get a bit of experience writing and running MapReduce code, and performing some follow-up analysis. In general, you will install and setup Hadoop and MapReduce, write a basic word count program, and then answer some questions about data flow in MapReduce applications.

General Homework Requirements

1. **Work Environment:** This homework must be written in Java.
2. **Programming:** Relevant tutorials and resources are linked below to aid in the programming portion of this homework.
3. **Academic Integrity: You will get an automatic F for the course if you violate the academic integrity policy.**
4. **Teams:** This homework is **an individual assignment**. You are not permitted to work with anyone else on this assignment. All work submitted must be yours and yours alone.

Submission Format

Your submission should be a single zip file that contains a src/ directory for your code, and a single PDF report. Submissions must be made on UBLearns by 4/3/23 @ 11:59PM. **No late submissions will be accepted. We will release the answer key before the midterm exam on 4/5/23.**

- Your src/ directory should contain your .java file, your input dataset (10 txt files from project gutenberg), and your list of stopwords, as well as any other files needed by your application (for example the list of patterns to skip).
- Your report should contain the answers to each of the 6 questions above, including any required screenshots of your application output.

Setup

To prepare your development environment for this homework you must first install and setup Java and Hadoop. **Make sure to start this process as early as possible, issues WILL come up.** To setup Hadoop, follow the instructions for pseudo-distributed operation here: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html

If you do not have a compatible Java SDK installed, you will need to install that now as well. Recommended Java installations are listed at the above link as well.

Test out your installation on the provided examples to make sure it is working properly before moving on with the rest of the assignment.

Once you have Hadoop setup, you must also get the data you will be using for the homework. Go to Project Gutenberg (https://www.gutenberg.org/ebooks/) and download 10 different books as plain text files. Each of the files you download must be at least 100 kB in size, but the larger the book the more interesting your results may be. Once you have the books downloaded, add them to your Hadoop instance so they will be readable by your MapReduce application.

Basic Word Count

Now that you have have setup Hadoop and downloaded your data, it is time to write your application. You can follow the tutorial for writing a WordCount application here: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html or use the demo from class as a starting point. Your application should be able to generate an aggregated list of word counts from all 10 books you downloaded.

Once you have your code working, run your code, and report the 25 most common words among the books you have chosen. Your list should not include any punctuation (other than punctuation that is actually part of the word), and should be case insensitive (Dog and dog would be counted as the same word).

Note that the output from MapReduce is sorted by key, not value. So to get the 25 most common words, you must sort the output yourself in whatever manner you see fit. The unix sort command would be one such way to do so.

Extending Your Application

You will likely notice that many of the words in your top 25 are extremely common words like "a" and "the" that don't have much semantic impact. These are commonly referred to as stop words, and are usually omitted from any kind of Natural Language Processing (NLP) analysis.

Extend your application to accept a list of stopwords and ignore those words in the mapping phase. Then re-run your application and provide a list of at least 50 common English stop words (you can find many such lists online), and report the 25 most common words.

Analysis

Answer the following questions based on your WordCount application. **For questions 1-4, include a screenshot of your application output that shows where your answer comes from in your report**.

1. What are the 25 most common words and the number of occurrences of each when you do not remove stopwords?
   Depends on your dataset

2. What are the 25 most common words and the number of occurrences of each when you do remove stopwords?
   Depends on your dataset

3. Based on the output of your application, how does removing stop words affect the total amount of bytes output by your mappers? Name one concrete way that this would affect the performance of your application.
   Removing/Omitting stop words reduces the total amount of bytes output by the mappers, because they are now emitting fewer key-value pairs. (note that the exact difference in bytes output can vary widely, depending on how commonly the stopwords appear in your input).
   The reduction in intermediate data will improve performance by reducing the amount of file I/O required, and the amount of data sent across the network.

4. Based on the output of your application, what is the size of your keyspace with and without removing stopwords? How does this correspond to the number of stopwords you have chosen to remove?
   Exact size of keyspace will vary based on your input dataset, but the difference in keyspace size after removing X stopwords will be X (assuming all of the stopwords appear at least once in your input set).

5. Let's now assume you were going to run your application on the entirety of Project Gutenberg. For this question, assume that there are **100TB** of input data, the data is spread over **10 sites**, and each site has **20 mappers**. Assume you ignore all but the **25 most common words** that you listed in question 2. Furthermore, assume that your combiners have been run optimally, so that each combiner will output at most 1 key-value pair per key.
   a. How much data will each mapper have to parse?
      (100TB/ 10 sites) / 20 mappers = .5TB
   b. What is the size of your keyspace?
      25
   c. What is the maximum number of key-value pairs that could be communicated during the barrier between mapping and reducing?
      25 keys per mapper * 20 mappers per site * 10 sites = 5,000

d. Assume you are running one reducer per site. On average, how many key-value pairs will each reducer have to handle?
This wording was a bit ambiguous. Each of the 10 total reducers will have to handle on average 2.5 keys worth of data (since there are 25 teams). This data comes from 5,000 key value pairs, so each reducer on average will handle on average 500 key value pairs worth of data. Note: it's still more like 2.5 key value pairs, where the key is the word, and the value is the list of counts found by all of the mappers. Either answer will be accepted here.

6. Draw the data flow diagram for question 5. The diagram should be similar to the diagram shown in lecture. On your diagram, label the specific quantities you got for 5a,b,c, and d.