Homework #2 - Spark

Due: 11/10/2024

Content Covered

Spark, Graph Algorithm

Homework Overview:

This assignment will provide you with hands-on experience in writing and executing Spark code, as well as conducting subsequent analysis. You will begin by installing and configuring Spark, then proceed to write fundamental programs including a word count program, Dijkstra's shortest path algorithm, and a page-rank algorithm. Following the implementation phase, you will be analyzing the data flow within Spark applications, gaining insights into how Spark processes and manipulates data across distributed computing environments. Through this comprehensive approach, you'll not only develop practical programming skills but also deepen your understanding of distributed data processing concepts using Spark.

General Homework Requirements:

- **Work Environment**: This homework can be written in Pyspark or Scala.
- **Programming**: You can use Jupyter Notebook, Jupyter Lab or Google collab , for Scala any IDE or Scala shell.
- **Academic Integrity**: You will get an automatic F for the course if you violate the academic integrity policy.
- **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:

1. Source Directory: All input data files, and code implementations should be organized within a specific directory named "src." This directory will contain both the input datasets and the code files required for the assignment.
2. Report: Prepare a comprehensive report containing answers to all questions posed in the assignment. Each answer should include suitable proofs or evidence to validate the authenticity of your submission and demonstrate that the outputs are legitimate. This report should be well-structured and provide clear explanations for each question, along with any necessary supporting materials ,documentation, and required screenshots .

By adhering to these refined requirements, you will ensure that your submission is well-organized, thoroughly documented, and adequately substantiated, thereby demonstrating your proficiency in completing the assignment successfully. **Failure to adhere to the specified submission format will result in a deduction of 3 marks. All submissions must follow the prescribed structure to ensure consistency and clarity. Submissions**

must be made on Brightspace by 10th November,2024 @ 11:59 PM. No late submissions will be accepted. Even a delay of 1 minute will result in a score of 0 for the assignment. It's recommended to upload your HW-2 at least a few hours before the due date and time to avoid any last-minute issues.

Setup:

- To prepare your development environment for this homework you must first install and set up PySpark. To install PySpark, follow the instructions here: https://spark.apache.org/docs/latest/api/python/getting_started/install.html
- Once you have Spark setup, you must also get the data you will be using for the homework. Go to Project Gutenberg (Project Gutenberg) and download 2 different books as plain text files.
- Run the code file "code_2.py" attached and you will get two text files named question2_1.txt and question2_2.txt.
- Run the code file "code_3.py" attached and you will get two text files named question3.txt.

Questions:

Part -1 Implement and analyze word count. Total 30 Marks

Implementation of basic word count 10 marks

You must write a basic word count program that reads two text files, counts the number of occurrences of each word in the text files, and outputs the result back to a text file as a list of key-value pairs, where the key is the word, and the value is the number of times the word occurred in the text file named output_1.txt. Provide screen shot of your output.

In addition to basic word counting, extend your code, which must also do the following:

- It must be case insensitive (see **lower()** in Python)
- It must ignore all punctuation (see, for example, **translate()** in Python)
- It must ignore stop words (see **filter()** in Spark)
- The output must be sorted by count in descending order (see **sortBy()** in Spark)

To accomplish this, you will use a combination of basic Python and RDD operations in PySpark or Scala. The following programming guide goes over basics of getting started with Spark and should contain everything you need to complete this part of the homework: https://spark.apache.org/docs/latest/rdd-programming-guide.html

Analysis

Answer the following questions based on your WordCount application. [2X10=20 marks]

1. In the PySpark REPL, run your extended word count program on a single text file.
   a. What are the 25 most common words? Include a screenshot of the program output to back up your claim.
   b. How many stages is execution broken up into? Explain why. Include a screenshot of the DAG visualization from Spark's WebUI to back-up your claim.

Part-2 Implement and analyze Dijakstra's Shortest Path algorithm. [4X10=40 Marks]

1. You must write a basic Dijakstra's shortest path algorithm for the two text files that were generated (question2_1.txt and question2_2.txt). Where the first column of each row is the initial node, the second column of each row is the destination, and the third column is the weight associated with the connection.
2. The algorithm should read both the files and compute the shortest path between the first node to all the other nodes and save them in a text file named output_2.txt.
3. Find the nodes with the greatest and the least distance from the starting node and provide a screenshot of the same.
4. How many stages is execution broken up into? Explain why. Include a screenshot of the DAG visualization from Spark's WebUI to back-up your claim.

Part-3 Implement and analyze Page-rank algorithm. [10X3=30 Marks]

1. You must write a basic page-rank algorithm considering the text file that is generated (question3.txt). It is a simulated network of 100 pages and its hyperlink .
   The algorithm should take the network provided and evaluate the page rank for all the webpages or nodes.
2. Find the node with the highest and the lowest page rank and provide a screenshot of the same.
3. How many stages is execution broken up into? Explain why. Include a screenshot of the DAG visualization from Spark's WebUI to back-up your claim.