CSE 587 Data Intensive Computing
Homework 2

Spring 2025

Due Date: Monday, April 7, 2025 by 11:59 PM

Topics: Distributed Processing and Spark Applications

This assignment is designed to introduce you to distributed data processing using Apache Spark, covering fundamental concepts of distributed processing and data-intensive computing, core Spark abstractions including Resilient Distributed Datasets (RDDs) and DataFrames, practical implementation of basic algorithms such as Word Count and PageRank, real-time data processing with Spark Streaming, and optimization techniques including an understanding of Spark's execution model (DAG, stages).

Learning Objectives:

- Understand the principles of distributed computing and how they are implemented in Spark.
- Gain hands-on experience with Spark RDDs by implementing a basic word count program.
- Develop proficiency in applying data transformation and actions in Spark.
- Implement and analyze the PageRank algorithm to understand link analysis in distributed settings.
- Learn to use Spark Streaming for real-time data processing tasks such as k-mer counting.
- Interpret Spark's WebUI, including DAG visualization and stage execution details.
- Enhance your ability to optimize and debug distributed data processing jobs.

General Requirements

- **Work Environment**: This homework must be written **in Python**. You can use Jupyter Notebook (.ipynb) or plain python script (.py). You should explore and read the documentation for using the related libraies.
- **Submission Format**: You will submit a single zip file containing:
  1. A comprehensive report (as a PDF) detailing your work, code, explanation, and results. (**Note: Every subtask in the assignment needs to be mentioned in the report. Write clearly about your key implementation steps, post your outputs, give detailed explanations and analysis.**)
  2. A folder named `src` containing:
     `book1.txt`
     `book2.txt`
     `word_count.py` or `word_count.ipynb`
     `output_1.txt`
     `output_1_extended.txt`
     `pagelinks.txt`
     `page_rank.py` or `page_rank.ipynb`
     `sentences.txt`
     `k-mer_count.py` or `k-mer_count.ipynb`

Your zip file should be named as cse587_hw2_UBIT_PersonNumber
(for example: cse587_hw2_abcd_12345678). **Failure to adhere to the specified submission format will result in a deduction of 5 points.** All submissions must follow the prescribed structure to ensure consistency and clarity.

- **Grading Criteria**: Correctness and completeness of tasks; Clarity of explanations and visualizations; Code readability and documentation.
- **Academic Integrity**: You will get an automatic F for the course if you violate the academic integrity policy. 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.

Notes:

- Late submissions are NOT allowed, therefore, you are encouraged to start working on Homework early on and submit well in time.
- If we are not able to run your code then **you will get -10 for this assignment**. Therefore, you must make sure that the code is executable not only on your device but by anyone having your code.
- AI violation policy will be strictly implemented and code/reports will be checked.

Part 1: Spark basics (60 points)

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

Task 1: Word count (30 points)

1. (8 points) 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.
2. (6 points) 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; search for what are stop words and design a stop word set yourself)
   - The output must be sorted by count in descending order (see sortBy() in Spark)

   Then, do step 1 again. (output file named `output_1_extended.txt`)
3. (4 points) Compare the two results and explain their differences in the report.
4. (12 points) Run your extended word count program on a single text file (one of the books) and answer these in the report:
   - (6 points) What are the 25 most common words? Include a screenshot of the program output to back up your claim.
   - (6 points) 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.
     (Hint: https://spark.apache.org/docs/3.5.3/web-ui.html)

Tips before starting

1. Install and setup PySpark (https://spark.apache.org/docs/latest/api/python/getting_started/install.html)
2. Download two different books as plain text files from https://www.gutenberg.org/ebooks/

Task 2: PageRank (30 points)

1. (10 points) Write a basic PageRank algorithm to analyze the generated text file `pagelinks.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. (10 points) Find the node with the highest and the lowest page rank, print them, and include a screenshot in your report.
3. (10 points) 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.

Tips before starting

1. Install and setup PySpark (https://spark.apache.org/docs/latest/api/python/getting_started/install.html)
2. Run the code file `generate_pagelinks.py` attached and you will get a text file named `pagelinks.txt`.

Part 2: Spark Streaming (40 points)

Task: k-mer count

1. (10 points) Create a local StreamingContext with two execution threads and a batch interval of 10 seconds.
2. (10 points) Create a DStream that represents streaming data from a TCP source (localhost:9999). (Hint: use Netcat)
3. (10 points) Generate k-mers of length 3 from each line of text.
4. (10 points) Count the occurrences of each k-mer and print the k-mer counts. Take screenshots of your output.

Tips before starting

1. Install and setup PySpark (https://spark.apache.org/docs/latest/api/python/getting_started/install.html)
2. Install SparkConf, SparkContext, and StreamingContext.
3. Install netcat (https://nmap.org/download.html)
4. Run the Python file named "sentence_generator.py" which will generate a text file named `sentences.txt`.
5. Learn how to use streaming in pyspark. (http://spark.apache.org/docs/latest/streaming-programming-guide.html)
6. You must write a k-mer counting program that will count the occurrences of k-mers in the text of `sentences.txt`.
7. For continuous streaming input, you will use `sentences.txt` as input stream through the TCP socket port 9999.

Bonus: Paper review (5 points)

Different from the bonus part in HW1 (where the bonus points are intended to help compensate for any lost points in other parts of the assignment, but the total score cannot exceed 100), in HW2, this bonus section has a weightage of 5% and will be added to your Final Grade. Requirements:

- Search Google Scholar for a paper related to Data Intensive Computing. (Hint: You can use any key words appeared in this course.)
- Write a comprehensive review of this paper. (400-800 words)
- The review should include:
  1. Why this paper is related to DIC?
  2. Your understanding of this paper.
  3. At least two main contributions and at least two limitations in this paper.
- **Usage of any Artificial Intelligence tools to generate text will result in an automatic 0 in this whole assignment and an additional -10 on your Final Grade.**
- **Your bonus review should be included at the end of your PDF report**