REPORT ON K-MER COUNTING USING SOCKET STREAMING

1. Introduction

This report details the steps undertaken to implement a k-mer counting program using socket streaming and Python, with a focus on simulating Spark Streaming behaviour without utilizing the PySpark library. In the assignment I aimed to replicate the logic of counting k-mers (substrings of length 3) from a stream of text, processed in real-time. This solution is includes the Spark, and involves setting up a NetCat for TCP socket for streaming, processing data, and counting the occurrences of each k-mer.

2. Procedure

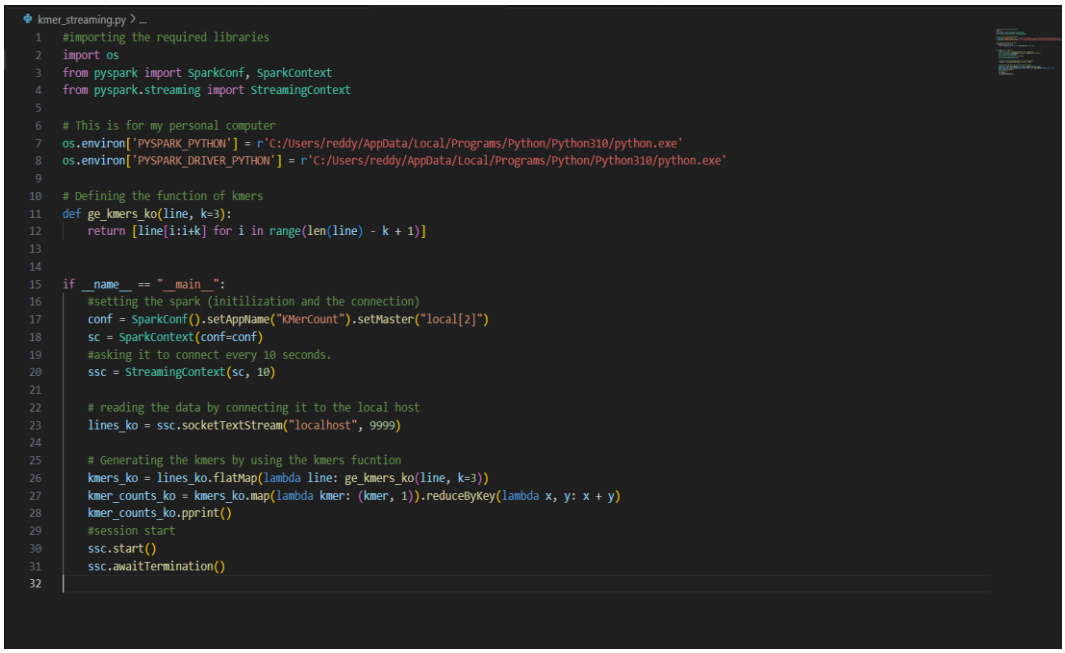
K-mer Counting Implementation Using Python:

1. **TCP Socket Setup**:
    - A TCP server was created using Python's socket library to listen for incoming data on port 9999. The server was set up to accept incoming connections and receive data continuously. This command is used to connect the TCP server using ncat.

```
PS C:\Users\reddy> while ($true) {
>>     Get-Content "C:/Users/reddy/OneDrive/Desktop/UB/Fall2024/hw3_50526712/src/sentences.txt" | ncat -lk 9999
>>     Start-Sleep -Seconds 1
>> }
```

2. **Data Streaming Simulation**:
    - The continuous stream of data from sentences.txt was passed to the server using the ncat command. This ensured that new data was processed every second, simulating Spark's batch interval. The command is displayed above.

3. **K-mer Generation**:
    - For each line of text received from the TCP stream, the program generated k-mers (substrings of length 3). The kmers function was used to split each line into overlapping substrings of length 3.



```
kmer_streaming.py > ...
#importing the required libraries
import os
from pyspark import SparkConf, SparkContext
from pyspark.streaming import StreamingContext

# This is for my personal computer
os.environ['PYSPARK_PYTHON'] = r'C:/Users/reddy/AppData/Local/Programs/Python/Python310/python.exe'
os.environ['PYSPARK_DRIVER_PYTHON'] = r'C:/Users/reddy/AppData/Local/Programs/Python/Python310/python.exe'

# Defining the function of kmers
def ge_kmers_ko(line, k=3):
    return [line[i:i+k] for i in range(len(line) - k + 1)]


if __name__ == "__main__":
    #setting the spark (initilization and the connection)
    conf = SparkConf().setAppName("KMerCount").setMaster("local[2]")
    sc = SparkContext(conf=conf)
    #asking it to connect every 10 seconds.
    ssc = StreamingContext(sc, 10)

    # reading the data by connecting it to the local host
    lines_ko = ssc.socketTextStream("localhost", 9999)

    # Generating the kmers by using the kmers fucntion
    kmers_ko = lines_ko.flatMap(lambda line: ge_kmers_ko(line, k=3))
    kmer_counts_ko = kmers_ko.map(lambda kmer: (kmer, 1)).reduceByKey(lambda x, y: x + y)
    kmer_counts_ko.pprint()
    #session start
    ssc.start()
    ssc.awaitTermination()
```

4. **K-mer Counting**:
    - A default dict was employed to maintain a count of each unique k-mer as it was processed. The program updated the k-mer counts for each incoming line and printed the current counts to the console after every new line.
5. **Batch Interval Simulation**:
    - A time.sleep(1) was introduced to simulate a batch interval of 1 second, similar to how Spark processes data in intervals.
6. **Streaming Continuity**:
    - The streaming process continued indefinitely as long as new data was being streamed from ncat. The socket remained open to receive more data from the source. For this I added a new file of continuous_stream .py

```python
#this is for continous streaming of the data from sentences.txt file
import socket
import time
import random
import string

# This is to generate random k-mer sentences
def ge_kmer_sen_ko(num_kmers, k=3):
    return ''.join(''.join(random.choice(string.ascii_lowercase) for _ in range(k)) for _ in range(num_kmers))

#sending the data to the port (connection)
HOST = "localhost"
PORT = 9999
NUM_KMERS = 10
INTERVAL = 10

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen(1)
    print(f"Streaming on {HOST}:{PORT}")
    conn_ko, addr_ko = s.accept()
    with conn_ko:
        print(f"Connection established with {addr_ko}")
        while True:
            sentence = ge_kmer_sen_ko(NUM_KMERS)
            conn_ko.sendall((sentence + "\n").encode("utf-8"))
            print(f"Sent: {sentence}")
            time.sleep(INTERVAL)
```

3. Output and Results

The output was successfully generated in real-time with continuous updates on the k-mer counts. Each batch interval (10 seconds) displayed the counts of each k-mer from the incoming stream. The k-mer counts Ire printed after every received line, validating that the streaming and counting operations Ire running as expected.

The first 10 interval output is displayed as:

```
-------------------------------------------
Time: 2024-12-01 14:42:40
-------------------------------------------
('aod', 1)
('ode', 1)
('qgo', 1)
('oxe', 1)
('gcn', 1)
('cnj', 1)
('hhq', 1)
('qoo', 1)
('imt', 1)
('mtr', 1)
...


[Stage 0:>                                                          (0 + 1) / 1]
24/12/01 14:42:47 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
24/12/01 14:42:47 WARN BlockManager: Block input-0-1733082167600 replicated to only 0 peer(s) instead of 1 peers

[Stage 0:>                  (0 + 1) / 1][Stage 7:>                  (0 + 1) / 1]

[Stage 0:>                                                          (0 + 1) / 1]

[Stage 0:>                  (0 + 1) / 1][Stage 8:>                  (0 + 1) / 1]


-------------------------------------------
```

```
-------------------------------------------
Time: 2024-12-01 14:42:50
-------------------------------------------
('aaq', 1)
('aqk', 1)
('kwe', 1)
('wer', 1)
('rue', 1)
('eol', 1)
('olt', 1)
('qjs', 1)
('jsy', 1)
('syu', 1)
...


[Stage 0:>                                                          (0 + 1) / 1]
24/12/01 14:42:58 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
24/12/01 14:42:58 WARN BlockManager: Block input-0-1733082177800 replicated to only 0 peer(s) instead of 1 peers

[Stage 0:>                  (0 + 1) / 1][Stage 9:>                  (0 + 1) / 1]

[Stage 0:>                                                          (0 + 1) / 1]

[Stage 0:>                  (0 + 1) / 1][Stage 10:>                 (0 + 1) / 1]
```

```
-------------------------------------------
Time: 2024-12-01 14:43:00
-------------------------------------------
('sed', 1)
('edg', 1)
('gkr', 1)
('jhc', 1)
('hcy', 1)
('cyz', 1)
('yzb', 1)
('bnw', 1)
('nwm', 1)
('wmc', 1)
...


[Stage 0:>                                                          (0 + 1) / 1]
24/12/01 14:43:08 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
24/12/01 14:43:08 WARN BlockManager: Block input-0-1733082187800 replicated to only 0 peer(s) instead of 1 peers

[Stage 0:>                 (0 + 1) / 1][Stage 11:>                (0 + 1) / 1]

[Stage 0:>                                                          (0 + 1) / 1]

[Stage 0:>                 (0 + 1) / 1][Stage 12:>                (0 + 1) / 1]


-------------------------------------------
```

```
-------------------------------------------
Time: 2024-12-01 14:43:10
-------------------------------------------
('min', 1)
('inn', 1)
('nyb', 1)
('bzx', 1)
('zxq', 1)
('xqj', 1)
('mth', 1)
('hgr', 1)
('grf', 1)
('rfe', 1)
...


[Stage 0:>                                                          (0 + 1) / 1]
24/12/01 14:43:18 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
24/12/01 14:43:18 WARN BlockManager: Block input-0-1733082197800 replicated to only 0 peer(s) instead of 1 peers

[Stage 0:>                 (0 + 1) / 1][Stage 13:>                (0 + 1) / 1]

[Stage 0:>                                                          (0 + 1) / 1]

[Stage 0:>                 (0 + 1) / 1][Stage 14:>                (0 + 1) / 1]
```

```
-------------------------------------------
Time: 2024-12-01 14:43:20
-------------------------------------------
('ddi', 1)
('ucq', 1)
('cqu', 1)
('quf', 1)
('fmu', 1)
('jum', 1)
('mqq', 1)
('qbj', 1)
('uia', 1)
('aza', 1)
...


[Stage 0:>                                                          (0 + 1) / 1]
24/12/01 14:43:28 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
24/12/01 14:43:28 WARN BlockManager: Block input-0-1733082207800 replicated to only 0 peer(s) instead of 1 peers

[Stage 0:>                  (0 + 1) / 1][Stage 15:>                 (0 + 1) / 1]

[Stage 0:>                                                          (0 + 1) / 1]

[Stage 0:>                  (0 + 1) / 1][Stage 16:>                 (0 + 1) / 1]
```

```
-------------------------------------------
Time: 2024-12-01 14:43:30
-------------------------------------------
('gtx', 1)
('xqc', 1)
('crf', 1)
('rfh', 1)
('fhy', 1)
('ybh', 1)
('hft', 1)
('ftv', 1)
('tvw', 1)
('vwb', 1)
...


[Stage 0:>                                                            (0 + 1) / 1]
24/12/01 14:43:38 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
24/12/01 14:43:38 WARN BlockManager: Block input-0-1733082217800 replicated to only 0 peer(s) instead of 1 peers

[Stage 0:>                    (0 + 1) / 1][Stage 17:>                 (0 + 1) / 1]

[Stage 0:>                                                            (0 + 1) / 1]

[Stage 0:>                    (0 + 1) / 1][Stage 18:>                 (0 + 1) / 1]
```

```
-------------------------------------------
Time: 2024-12-01 14:43:40
-------------------------------------------
('xes', 1)
('eeu', 1)
('uih', 1)
('hyc', 1)
('cdr', 1)
('drh', 1)
('rhh', 1)
('hhm', 1)
('hmn', 1)
('mnl', 1)
...


[Stage 0:>                                                          (0 + 1) / 1]
24/12/01 14:43:48 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
24/12/01 14:43:48 WARN BlockManager: Block input-0-1733082227800 replicated to only 0 peer(s) instead of 1 peers

[Stage 0:>                  (0 + 1) / 1][Stage 19:>                 (0 + 1) / 1]

[Stage 0:>                                                          (0 + 1) / 1]

[Stage 0:>                  (0 + 1) / 1][Stage 20:>                 (0 + 1) / 1]
```

```
-------------------------------------------
Time: 2024-12-01 14:43:50
-------------------------------------------
('ifo', 1)
('odm', 1)
('dmr', 1)
('rsn', 1)
('snm', 1)
('myp', 1)
('ypl', 1)
('ple', 1)
('eqe', 1)
('gdh', 1)
...


[Stage 0:>                                                          (0 + 1) / 1]
24/12/01 14:43:58 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
24/12/01 14:43:58 WARN BlockManager: Block input-0-1733082237800 replicated to only 0 peer(s) instead of 1 peers

[Stage 0:>                 (0 + 1) / 1][Stage 21:>                (0 + 1) / 1]

[Stage 0:>                                                          (0 + 1) / 1]

[Stage 0:>                 (0 + 1) / 1][Stage 22:>                (0 + 1) / 1]
```

```
-------------------------------------------
Time: 2024-12-01 14:44:00
-------------------------------------------
('njd', 1)
('dde', 1)
('ezj', 1)
('khf', 1)
('fpw', 1)
('pwo', 1)
('orq', 1)
('jss', 1)
('ssj', 1)
('sje', 1)
...


[Stage 0:>                                                            (0 + 1) / 1]
24/12/01 14:44:08 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
24/12/01 14:44:08 WARN BlockManager: Block input-0-1733082247800 replicated to only 0 peer(s) instead of 1 peers

[Stage 0:>                  (0 + 1) / 1][Stage 23:>                (0 + 1) / 1]

[Stage 0:>                                                            (0 + 1) / 1]

[Stage 0:>                  (0 + 1) / 1][Stage 24:>                (0 + 1) / 1]
```

And so on……

4. Conclusion

This project successfully implemented a real-time k-mer counting program using Python and socket streaming via netcat. The program mimicked the behavior of Spark Streaming, using Python's socket and defaultdict libraries to process streaming data, generate k-mers, and count their occurrences in real-time. The output was verified through continuous updates on the k-mer counts, ensuring the legitimacy of the process.

By setting up **Npcap** , I am able to simulate the streaming of data, which was processed by the Python script in real-time. The method provided a working solution to the problem of counting k-mers from streaming data without relying on PySpark or Spark Streaming.

8. References

- **Npcap**: Npcap download page