Assignment 3

Anurag Nagar, Ph.D.

Due Date: Mentioned on eLearning

Instructions

- This assignment requires you to write code using Structured Streaming and GraphFrames libraries. To be able to run your code, you would need Apache Spark and other libraries mentioned installed in your environment. If you choose to run it on your local machine, all paths should be inputted as parameters. You cannot hardcode paths to your local computer.
- All instructions for compiling and running your code must be placed in the README file.
- You should use a cover sheet, which can be downloaded from http://www.utdallas.edu/~axn112530/cs6350/CS6350_CoverPage.docx
- You are allowed to work in pairs i.e. a group of two students is allowed. Please write the names of the group members on the cover page.
- **You have a total of 4 free late days for the entire semester. You can use at most 2 days for any one assignment. After four days have been used up, there will be a penalty of 10% for each late day. The submission for this assignment will be closed 2 days after the due date.**
- Please ask all questions on Piazza, not via email.

1 Spark Streaming with Real Time Data and Kafka

In this part, you will create a Spark Streaming application that will continuously read text data from a real time source, analyze the text for named entities, and send their counts to Apache Kafka. A pipeline using Elasticsearch and Kibana will read the data from Kafka and analyze it visually.

Setting up your development environment

You can set up everything on your local machine or host them on the cloud. Following are the resources that you will need:

- You will need to get access to a real-time text data source. Some examples are:
  - **Reddit** - You can access the comments on a subreddit using the PRAW Python library: https://praw.readthedocs.io/en/stable/
    There is also a streaming API for this at:
    https://praw.readthedocs.io/en/latest/code_overview/other/subredditstream.html.
    You are free to choose any subreddit.
  - **NewsAPI** - You can download real time news using the News API available at: https://newsapi.org/. There is also a Python wrapper for this available at: https://newsapi.org/docs/client-libraries/python
  - **Finnhub** For financial news and analysis, the Finnhub library https://finnhub.io/ can be used. There is a Python wrapper for this available at https://pypi.org/project/finnhub-python/.
- Download Apache Kafka and go through the quickstart steps:
  `https://kafka.apache.org/quickstart`
- Windows users might want to consider WSL, which gives a Unix-like environment on Windows:
  `https://docs.microsoft.com/en-us/windows/wsl/install-win10`
- You will need a working Spark cluster. It can be setup locally or you can use a cloud based server.
- I have attached a simple project showing how to pass data between Kafka and a Spark Structured Streaming application that performs word count. Instructions for running are also attached.
- Later, you will also need to set up Elasticsearch and Kibana environment to visualize data. You will need to download Elasticsearch, Kibana, and Logstash from
  `https://www.elastic.co/downloads/`

Project Steps

For this project, you will need to perform the following steps:

- Create a Python application that reads from a real-time data source, such as the ones mentioned in the previous part. Some libraries have a streaming version that continuously fetch real-time data. For others, you might have to write a loop that gets real time data at periodic intervals.

- This incoming data should continuously be written to a Kafka topic (let's call it topic1 for illustration).
- Create a PySpark structured streaming application that continuously reads data from the Kafka topic (topic1) and keeps a running count of the named entities being mentioned. You should already know how to extract named entities from text. In this part, you will keep a running count of the named entities being mentioned.
- At the trigger time, a message containing the named entities and their counts should be sent to another Kafka topic (let's call it topic2 for illustration).
- Configure Logstash, Elasticsearch, and Kibana to read from the Kafka topic (topic2) and create a bar plot of the top 10 most frequent named entities and their counts.

Visualizing the data using Elasticsearch and Kibana

You will need Elasticsearch and Kibana installed to get the data from Kafka and visualize it. Details are available at:

https://www.elastic.co/downloads
https://www.elastic.co/start

You will need to visualize the count of top 10 named entities being mentioned on your chosen data source. You should be able to view this using a barplot in Kibana.

What to Submit

You are required to submit the following:

- Your project code file.
- Named entity frequency bar plots taken at several intervals. For example, you can show the top 10 named entities being mentioned after 15, 30, 45, and 60 minutes.
- A report explaining your data source and also a small paragraph explaining what your results indicate.
- A README file indicating how to run your code. Please do not use any hard coded paths.

2 Analyzing Social Networks using GraphX/GraphFrame

In this part, you will use Spark GraphX/GraphFrame to analyze social network data. You are free to choose any one of the Social network datasets available from the SNAP repository.

You will use this dataset to construct a GraphX/GraphFrame graph and run some queries and algorithms on the graph. You will need to perform the following steps:

2.1 Loading Data

Load the data into a GraphFrame or RDD using Spark. Define a parser so that you can identify and extract relevant fields. Note that edges are directed, so if your dataset has undirected relationships, you might need to convert those into 2 directed relationships. That is, if your dataset contains an undirected friendship relationship between X and Y, then you might need to create 2 edges one from X to Y and the other from Y to X.

2.2 Create Graphs

Define edge and vertex structure and create property graphs.

2.3 Running Queries

Run the following queries using the GraphX/GraphFrame API and write your output to a file on the cluster.

a. Find the top 5 nodes with the highest outdegree and find the count of the number of outgoing edges in each

b. Find the top 5 nodes with the highest indegree and find the count of the number of incoming edges in each

c. Calculate PageRank for each of the nodes and output the top 5 nodes with the highest PageRank values. You are free to define any suitable parameters.

d. Run the connected components algorithm on it and find the top 5 components with the largest number of nodes.

e. Run the triangle counts algorithm on each of the vertices and output the top 5 vertices with the largest triangle count. In case of ties, you can randomly select the top 5 vertices.

2.4 What to submit

You are required to submit the following:

- Your project code file.
- Output file with the results of the queries
- A brief summary indicating whether your results make sense and what insights you get by this analysis.
- A README file indicating how to run your code. Please do not use any hard coded local paths.