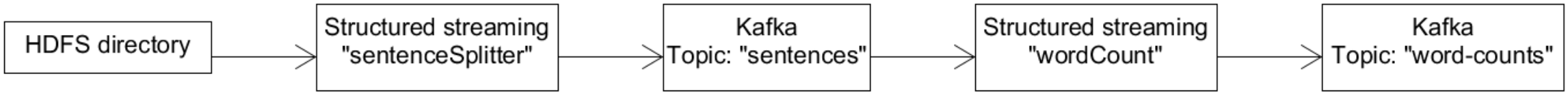
Lecture 6 exercises

Daniel Skjold Toft

Follow up on lecture 4

- What is Spark and what is Python? (Great question!)
  - Navigate to “BDDST21\Lecture4\sentimentExercise” and look at “myAnswer.py” and “wrong.py”
- The wrong collects the data and uses Python looping
- The other, creates a Python function, but uses Spark to map it
  - Utilizing distributed computing by Spark

The architecture for today



- For that, we need:
  - HDFS cluster
  - Spark cluster
  - Kafka and Zookeeper cluster (+ Kowl)
- Navigate to BDDST21\Lecture6
  - “docker-compose up”

Make the HDFS directory

- “docker exec –ti namenode bash”
- “hdfs dfs –mkdir /stream-in” – Create a directory in the HDFS

Sentence splitter – structured streaming

- Splits sentences based on text files in a directory
  - Read files from a HDFS directory
  - Split the content of file into sentences, write to Kafka topic
- Dockerfile has been changed this time!

- Navigate to BDDST21\Lecture6\sentenceSplitter
  - “run”

Download the book!

- “docker exec –ti namenode bash”
- “apt update”
- “apt install wget”
- “wget -O alice.txt https://www.gutenberg.org/files/11/11-0.txt”
- “hdfs dfs -put alice.txt /stream-in/”

Take a look at Kowl!

- Navigate to address “localhost:8088”
- Navigate to “topics”
- May take a minute or two before processing is done

Word count – structured streaming

- Count the words from the Kafka topic “sentences”
  - Create a Kafka read stream
  - Make the word count
  - Modify the DataFrame to be writeable to Kafka
  - Create the Kafka write stream
- Output mode is “complete”
  - Write all of the DataFrame everytime it receives updates

Put the book in again!

- The word count program starts from the latest Kafka message offset, meaning it will not “know” about the previous messages on the topic “sentences”
- “docker exec –ti namenode bash”
- “hdfs dfs -put alice.txt /stream-in/alice2.txt”
  - Uses a new name, to not overwrite the other

Take a look at Kowl!

- Navigate to “topics”
- May take a minute or two before processing is done

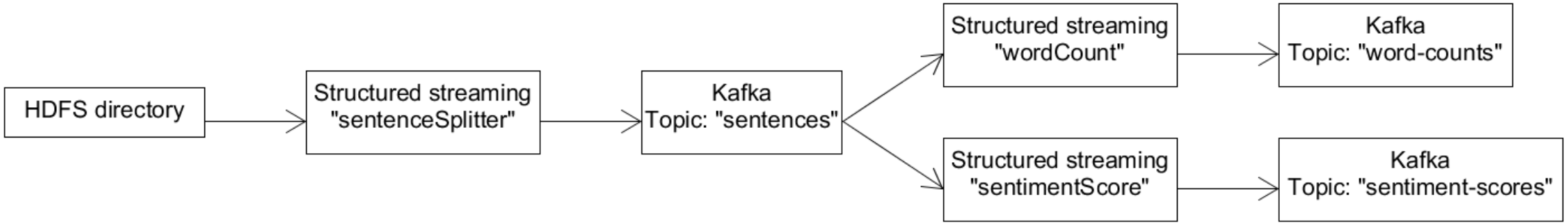
Your turn!

- What is the sentiment of each sentence?
- Read from the Kafka topic “sentences”
- Calculate the sentiment score of each sentence
- Sent it back to Kafka!

- Is Kafka the best way to store the results of word count and sentiment score?

Your turn!

- Final architecture:



Your turn!

- Final output should look something like:
  - {“sentimentScore”: 3, “count”: 7}
  - {“sentimentScore”: 2, “count”: 13}
  - {“sentimentScore”: 1, “count”: 30}
  - {“sentimentScore”: 0, “count”: 157}
  - {“sentimentScore”: -1, “count”: 19}
  - {“sentimentScore”: -2, “count”: 5}