IT UNIVERSITY OF COPENHAGEN
BIG DATA MANAGEMENT 2022

Assignment 2: Scalable Processing

Yelp Reviews and Authenticity

September 2022

1 INTRODUCTION

In this assignment you will be working with semi-structured data (JSON) on a Spark cluster.[1] The main task is to perform queries and basic analytics on a (somewhat) big data. The dataset is not huge, but big enough that working with it on a normal laptop would quickly become tiresome.

This assignment has two parts. In the first part, you will be asked to implement certain queries using the procedural methods of the Spark DataFrame API. In the second part you will be asked to do more in-depth data analysis using Spark.

Authenticity Study

In 2019, an interesting study looked into the use of the word "authentic" in Yelp reviews and how it was used to signal different characteristics from cuisine to cuisine. They found that, for example, in Chinese and Mexican restaurants, the word was used to describe typically negative things like "dirty", "kitsch" or "cheap", where Italian and French restaurants used to signal high quality and delicious details. The author sees this tendency as harmful, trapping non-white restaurant-owners in negative stereotypes.

The study was done by manually reading and classifying 20,000 reviews for restaurants in New York City. The dataset you will be using is vastly larger (6,685,900 reviews) and much geographically diverse. The second part of the assignment is too see if you can support or refute the claims of the article using this larger dataset and more automated techniques.

2 THE DATA

The data you will use is from the YELP ACADEMIC DATASET (documentation), a dataset of reviews and tips given by yelp users to business. The version of the dataset we are using takes up around 8 gigabytes uncompressed and includes 6,685,900 reviews.

The data is saved as JSON-files, which have been uploaded to the Hadoop Distributed File System (HDFS) that is running on the cluster. This means that the JSON files have been partitioned into chunks that are replicated and distributed across the machines in the cluster.

[1] To access the cluster, see the guide on LearnIT

Spark reads the files from HDFS and converts them to in-memory DataFrames.[2] As JSON is semi-structured, Spark makes two passes over the data, first determining which columns to include in the dataframe, then inserting the data.

The JSON-files are:

```
FULL_PATH_NAME                                  SIZE
/datasets/yelp/business.json                    131.9 M
/datasets/yelp/checkin.json                     389.9 M
/datasets/yelp/review.json                      5.0 G
/datasets/yelp/tip.json                         233.2 M
/datasets/yelp/user.json                        2.3 G
```

Besides the JSON-version of the dataset, the data is also stored as Parquet files, a column-oriented binary file format optimized for the Hadoop ecosystem. These files are placed in the "parquet"-subfolder:

```
FULL_PATH_NAME                                  SIZE
/datasets/yelp/parquet/business.parquet         21.4 M
/datasets/yelp/parquet/checkin.parquet          172.1 M
/datasets/yelp/parquet/review.parquet           2.8 G
/datasets/yelp/parquet/tip.parquet              105.9 M
/datasets/yelp/parquet/user.parquet             1.7 G
```

Reading files from HDFS in Spark is done with the `spark.read`-module, which has methods for different file formats.

```
# read a json file
df1 = spark.read.json("path_to_the_file.json")
# read a parquet file
df2 = spark.read.parquet("path_to_the_file.parquet")
```

3 Requirements and Hand-in

The assignment consists of two sections: In the first you will create specific queries using Spark DataFrames, while the second is more freeform where you will answer high-level questions by querying the data and reasoning the results.

The maximum length for the report handed in is five pages.

3.1 Specific DataFrame Queries

Formulate the following queries using Spark DataFrames, not using the `spark.sql`-method:

1. Analyze `business.json` to find the total number of reviews for all businesses. The output should be in the form of a Spark DataFrame with one value representing the count.
2. Analyze `business.json` to find all businesses that have received 5 stars and that have been reviewed by 1000 or more users. The output should be in the form of DataFrame of (name, stars, review count).
3. Analyze `user.json` to find the influencers who have written more than 1000 reviews. The output should be in the form of DataFrame of user id.
4. Analyze `review.json`, `business.json`, and a view created from your answer to Q3 to find the businesses that have been reviewed by more than 5 influencer users.
5. Analyze `review.json` and `user.json` to find an ordered list of users based on the average star counts they have given in all their reviews.

3.2 Authenticity Study

These questions should be answered using statistics found in the data. You are free to use the `spark.sql`-method for this section.

[2] Not the same as Pandas DataFrames, but inspired by it.

3.2.1 *Data Exploration*

Formulate Spark SQL queries to explore the use of "authenticity language", as defined in the eater.com article mentioned above. These queries should include (but not be limited to) the following questions:

- What is the percentage of reviews containing a variant of the word "authentic"? How many reviews contain the string "legitimate" grouped by type of cuisine?
- Is there a difference in the amount of authenticity language used in the different areas? (e.g., by state, north/south, urban/rural)
  Note: As part of answering this question, you should compute the full cube combining the location of the business and whether the review contains authenticity language, and use this to aggregate their counts per state and city.

Explain your exploratory approach. Explain the queries you formulate, the results you get and how they inform your further exploration.

3.2.2 *Hypothesis Testing*

The hypothesis proposed in the article is that authenticity language is used to describe different characteristics for different cuisines (and by extension, makes it harder for non-white restaurant owners to enter the higher-end restaurant market).

- Can you identify a difference in the relationship between authenticity language (words such as "authentic" or "legitimate" or their derived forms) and typically negative words (like "dirty", "kitsch", "cheap", "rude", "simple" or similar), in restaurants serving south american or south asian cuisine than in restaurants serving european cuisine? And to what degree?

Explain your approach, assumptions and results. Make an argument as to how your queries actually answer the question.

3.3 Bonus Question

Try starting a new shell and read the data from the parquet-file instead of the JSON file. Do you notice any difference in the efficiency? Why (not)? How about if you apply filters and do actions like `count` or `collect`?

3.4 Suggested reading and useful links

- Spark 2.3.1 Quick Start (Remember to choose the python tab)
- Spark SQL Programming Guide (Much more detail)
- Our cluster scheduler overview (See how many resources are free on the cluster)
- Architecture Overview of Cluster-based Spark
- The PySpark API Documentation for version 2.3.1 (Remember to always use the documentation for this version)
  Especially these modules will be useful to you:
  - DataFrame
  - Column
  - Spark SQL Functions
- A Tale of Three Apache Spark APIs: RDDs vs DataFrames and Datasets

4 Code Example

```
# This import is always needed
import pyspark

#####################################################
#
#            ONLY USE BELOW CODE IF
#     YOU RUN YOUR CODE USING SCRIPTS LIKE:
#      pyspark < your_script_file_task1.py
#
#####################################################
# Only alter these configurations if strictly needed.
# If you have memory errors, you ask spark for more resourses.
# By default you're allocated enough resources for most tasks.

from pyspark.sql import SparkSession
from pyspark import SparkConf
conf = SparkConf()
conf.set("spark.executor.memory", "1G")   # 1 Gigabyte...
conf.set("spark.executor.instances", "2") # ...for 2 executors = 2 GB total
# And you can add other configuration options here.

# The "spark" variable needed for most things. You should change the
    appName
spark = SparkSession.builder \
                    .appName('template-application') \
                    .config(conf=conf) \
                    .getOrCreate()
#####################################################
#
#            ABOVE CODE IS NOT FOR NOTEBOOKS/
#                  INTERACTIVE SHELL
#       ("spark" variable created automatically)
#
#####################################################


# Insert your own code here

########################################
# BELOW CODE CAN BE USED
# NO MATTER HOW YOU RUN YOUR CODE :)
########################################

# Example: Say we're only interested in reviews of good mexican restaurants
     in Arizona. You can delete this when you do your own thing.

# Read in the business and review files
bs = spark.read.json("/datasets/yelp/business.json")
rs = spark.read.json("/datasets/yelp/review.json")

# Get number of rows:
rs.count()

# Reduce resource usage and make queries run faster
# by only using a small sample of the dataframe
# and overwriting previous variable "df".
# Very useful while developing, not so much to
# provide final answers. Therefore: Remember to
# to re-read the df when done developing code using
# df = spark.read etc like above.
rs = rs.sample(withReplacement=False, fraction=1/100)
rs.count()
```

```
61
62 # Filter to only Arizona businesses with "Mexican" as part of their
       categories
63 az_mex = bs.filter(bs.state == "AZ").filter(bs.categories.rlike("Mexican"))
       .select("business_id", "name")
64
65 # Join with the reviews
66 az_mex_rs = rs.join(az_mex, on="business_id", how="inner")
67
68 # Filter to only 5 star reviews
69 good_az_mex_rs = az_mex_rs.filter(az_mex_rs.stars == 5).select("name","text
       ")
70
71 # Print the top 20 rows of the DataFrame
72 good_az_mex_rs.show()
73
74 #Convert to pandas (local object) and save to local file system (ambari0)
75 good_az_mex_rs.toPandas().to_csv("good_az_reviews.csv", header=True, index=
       False, encoding='utf-8')
```