Project #2 Big Data Tools

Gabriel de O. F. Gonçalves (uc2023238703)   João Pereira (uc2023243538)

Faculdade de Ciências e Tecnologia da Universidade de Coimbra

IACD - MECD

1 Kafka (Assigment 1)

- **Before any task, start Zookeper and Kafka:**
```
sudo systemctl start zookeeper
sudo systemctl start kafka
```

1.1 Task1

- **Create a kafka topic called task1-topic with one partition:**
```
kafka-topics --create --topic task1-topic --bootstrap-server localhost:9092 --partitions
1 --replication-factor 1
```

1.1.1 Kafka producer

- **Initialize the Kafka producer:**
```
producer = KafkaProducer( bootstrap_servers='localhost:9092',
value_serializer=lambda v: json.dumps(v).encode('utf-8') )
```

- **Define the random data:**
```
sensor_id = 1 temperature_reading = "sensor_id": sensor_id,
"temperature": round(random.uniform(15.0, 35.0), 1)
```

- **Send Data to Kafka Topic:**
```
producer.send('task1-topic', value=temperature_reading)
print(f"Sent: temperature_reading")
```

1.1.2 Kafka consumer

- **Set up Kafka consumer**:
```
consumer = KafkaConsumer( 'task1-topic',
bootstrap_servers='localhost:9092',
value_deserializer=lambda x: json.loads(x.decode('utf-8')) )
```

- **Consumer listens to the topic and continuously retrieves messages:**
```
for message in consumer:
print(f"Received message: message.value")
```

1.2 Task 2

- **Create a kafka topic called task2-topic with 3 partitions:**
```
kafka-topics --create --topic task1-topic --bootstrap-server localhost:9092 --partitions
3 --replication-factor 1
```

1.2.1 Kafka producer

- **Initialize the Kafka producer:**
```
producer = KafkaProducer( bootstrap_servers='localhost:9092',
value_serializer=lambda v:  json.dumps(v).encode('utf-8') )
```

- **Define users and activities:**
```
user_ids = [f"useri" for i in range(1, 5)]
activities = ["login", "logout", "purchase", "browse", "signup"]
```

- **Generate and send user activity logs**:
```
activity_log = {
"user_id":  random.choice(user_ids),
"activity":  random.choice(activities) }
producer.send('task2-topic', value=activity_log)
print(f"Sent:  activity_log")
```

1.2.2 Kafka consumer

- **Set up Kafka consumer to listen to the topic task2-topic:**
```
consumer = KafkaConsumer( 'task2-topic',
bootstrap_servers='localhost:9092',
group_id=group_id,
auto_offset_reset='earliest',
value_deserializer=lambda v:  json.loads(v.decode('utf-8')) )
```

- **Consume and Process Messages:**
```
for message in consumer:
print(f"Consumer consumer_id received:  message.value")
```

1.2.3 Multiple activated consumers behavior

- **Three consumers in activity-group are active:**
```
Each consumer consumes from 1 partition
```

- **Two consumers in activity-groupare active:**
```
One consumer consumes from 2 partitions and the other consumes from the left partition
```

- **Four consumers in activity-groupare active:**
```
Only 3 consumers are able to consume and the 4th consumer dont consume from any partition
```

1.3 Task 3

- **Create Kafka Topics: Purchase and User-Activity:**
```
‘kafka-topics --create --topic purchase-topic --bootstrap-server localhost:9092 --partitions
1 --replication-factor 1‘
‘kafka-topics --create --topic user-activity-topic --bootstrap-server localhost:9092
--partitions 1 --replication-factor 1‘
```

1.3.1 Kafka producer

- **Initialize the Kafka producer:**
```
producer = KafkaProducer( bootstrap_servers='localhost:9092',
value_serializer=lambda v:  json.dumps(v).encode('utf-8') )
```

- **Define user actions:**
```
items = ["book", "laptop", "phone", "headphones"]
activities = ["login", "page_view", "add_to_cart", "checkout"]
```

- **Send Data to Kafka Topics:**
```
producer.send('purchase-topic', purchase_data)
producer.send('user-activity-topic', activity_data)
```

1.3.2 Kafka consumer purchase-group and activity-group

- **Both use the same logic.**
- **Set up Kafka consumer to listen to the topic:**
```
consumer = KafkaConsumer( 'purchase-topic',
bootstrap_servers='localhost:9092',
group_id=group_id,
value_deserializer=lambda v:  json.loads(v.decode('utf-8')) )
```
- **Dictionary to keep track info:**
```
user_total_spent = defaultdict(float)
```
- **Consume and Process Messages:**
```
for message in consumer:
purchase = message.value
user_id = purchase["user_id"]
amount = purchase["amount"]

user_total_spent[user_id] += amount
print(f"User user_id Total Spent:user_total_spent[user_id]")
```

2 Hadoop (Assigment 2)

2.1 Task 1

- **Run task 1:**
```
python task1.py fakefriends.csv
```
- **Define a MRStep with the mapper and reducer:**
```
Mapper:  Processes each line of input data and emits age as the key and the number
of friends as the value.
Reducer:  Aggregates the number of friends by age and calculates the average.
```
- **def mapper_get_average(self, _, line):**
```
Splits each line of input into fields:  userId, userName, age, numFriends.
Yields the age as the key and the number of friends as the value.
```
- **def reducer_get_average(self, age, values):**
```
In each age value, sum the number of friends and count the number of people that
have that age.
Devide those two values to calculate the average of friends at a given age.
```

2.2 Task 2

- **Run task 2:**
```
python task2.py 1800.csv
```
- **Define a MRStep with the mapper and reducer:**
```
Mapper:  Emits the weather station ID as the key and its temperature as the value.
Reducer:  Calculates the minimum temperature for each weather station.
```

- **def mapper_get_min(self, _, line):**
  `Splits each input line into columns.`
  `Extracts the weather station ID (first column) and temperature (fourth column).`
  `Yields the weather station ID as the key and the temperature as the value.`

- **def reducer_get_min(self, weather_station, values):**
  `Receives all temperature values for each weather station.`
  `Calculates the minimum temperature.`
  `Yields the weather station ID and its minimum temperature.`

2.3 Task 3

- **Run Task 3:**
  `python task3.py Book`

- **Define two MRSteps, one with the mapper and reducer, the other with the reducer_sort**

- **def mapper(self, _, line):**
  `Processes each line of text.`
  `Extracts words using a regular expression.`
  `Emits each word with a count of 1.`

- **def reducer(self, word, counts):**
  `Receives all counts for each word.`
  `Sums the counts to calculate the total frequency.`
  `Emit total count and word as a tuple to prepare for sorting.`

- **def reducer_sort(self, _, word_count_pairs):**
  `Receives all (total_count, word) pairs as a list.`
  `Sorts the list in descending order of frequency.`
  `Emits each word and its frequency.`

2.4 Task 4

- **Run Task 4:**
  `python task4.py customer-orders.csv`

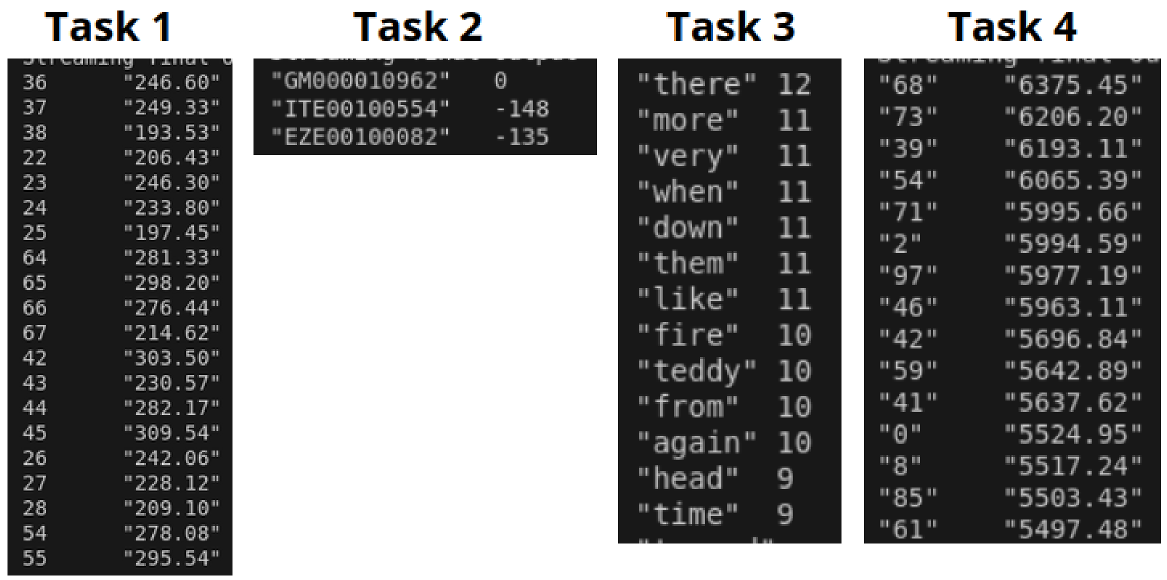
- **Define two MRSteps, one with the mapper and reducer_get_sum, the other with the reducer_sort**

- **def mapper(self, _, line):**
  `Reads each input line.`
  `Extracts customerId and amount.`
  `yield both values.`

- **def reducer_get_sum(self, customer_id, values):**
  `Receives all amounts for each customer.`
  `Calculates the total amount spent by summing the values.`
  `Emits a tuple (total_amount, customerId).`

- **def reducer_sort(self, _, customer_amount_pairs):**
  `Receives a list of (total_amount, customerId) tuples`
  `Sorts the list in descending order of total_amount Emits total amount spent serted by each customer.`

2.5 Terminal Outputs



```
36      "246.60"
37      "249.33"
38      "193.53"
22      "206.43"
23      "246.30"
24      "233.80"
25      "197.45"
64      "281.33"
65      "298.20"
66      "276.44"
67      "214.62"
42      "303.50"
43      "230.57"
44      "282.17"
45      "309.54"
26      "242.06"
27      "228.12"
28      "209.10"
54      "278.08"
55      "295.54"
```

```
"GM000010962"   0
"ITE00100554"   -148
"EZE00100082"   -135
```

```
"there" 12
"more"  11
"very"  11
"when"  11
"down"  11
"them"  11
"like"  11
"fire"  10
"teddy" 10
"from"  10
"again" 10
"head"  9
"time"  9
```

```
"68"    "6375.45"
"73"    "6206.20"
"39"    "6193.11"
"54"    "6065.39"
"71"    "5995.66"
"2"     "5994.59"
"97"    "5977.19"
"46"    "5963.11"
"42"    "5696.84"
"59"    "5642.89"
"41"    "5637.62"
"0"     "5524.95"
"8"     "5517.24"
"85"    "5503.43"
"61"    "5497.48"
```

Figure 1: Outputs for all tasks in the assignment 2.

3 Spark RDDs (Assigment 3)

3.1 Task 1

- Configure Spark application.
- Load dataset.
- Split each line by comma.
- Transform to key-value pairs.
- Find the minimum temperature for each category.
- Collect and print results.

3.2 Tasks 2 and 3

- Configure Spark application.
- Load the book text file as an RDD.
- Define the line number to start processing (zero-indexed).
- Filter RDD to keep only lines starting from the specified line.
- Split lines into words, convert to lowercase, and count word frequency.
- Sort word counts by frequency in ascending order.
- Collect and print results.

3.3 Tasks 4 and 5

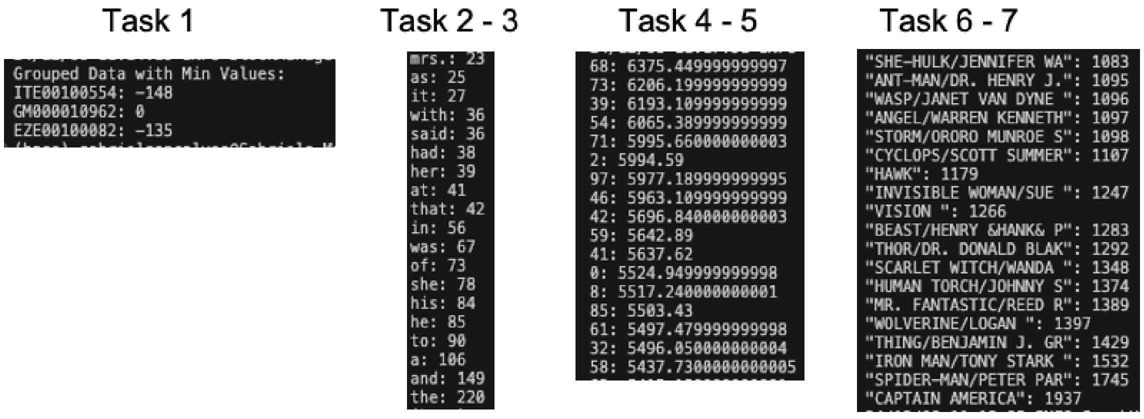
- Configure Spark application.
- Load the dataset (customer orders) as an RDD.
- Split each line by comma.
- Transform into key-value pairs (`Customer_ID, Amount`).

- Compute the total amount spent by each customer by summing values.
- Sort the results by total amount in descending order.
- Collect and print results.

3.4 Tasks 6 and 7

- Configure Spark application.
- Load the graph dataset as an RDD.
- Count the frequency of each superhero ID.
- Load the names dataset as an RDD.
- Map names dataset to `(ID, Name)` pairs.
- Join the two datasets on superhero ID.
- Map to `(Name, Frequency)` and sort by frequency in ascending order.
- Collect and display the results.

3.5 Terminal Outputs



```
Grouped Data with Min Values:
ITE00100554: -148
GM000010962: 0
EZE00100082: -135
```

```
mrs.: 23
as: 25
it: 27
with: 36
said: 36
had: 38
her: 39
at: 41
that: 42
in: 56
was: 67
of: 73
she: 78
his: 84
he: 85
to: 90
a: 106
and: 149
the: 220
```

```
68: 6375.449999999997
73: 6206.199999999999
39: 6193.109999999999
54: 6065.389999999999
71: 5995.660000000003
2: 5994.59
97: 5977.189999999995
46: 5963.109999999999
42: 5696.840000000003
59: 5642.89
41: 5637.62
0: 5524.949999999998
8: 5517.240000000001
85: 5503.43
61: 5497.479999999998
32: 5496.050000000004
58: 5437.7300000000005
```

```
"SHE-HULK/JENNIFER WA": 1083
"ANT-MAN/DR. HENRY J.": 1095
"WASP/JANET VAN DYNE ": 1096
"ANGEL/WARREN KENNETH": 1097
"STORM/ORORO MUNROE S": 1098
"CYCLOPS/SCOTT SUMMER": 1107
"HAWK": 1179
"INVISIBLE WOMAN/SUE ": 1247
"VISION ": 1266
"BEAST/HENRY &HANK& P": 1283
"THOR/DR. DONALD BLAK": 1292
"SCARLET WITCH/WANDA ": 1348
"HUMAN TORCH/JOHNNY S": 1374
"MR. FANTASTIC/REED R": 1389
"WOLVERINE/LOGAN ": 1397
"THING/BENJAMIN J. GR": 1429
"IRON MAN/TONY STARK ": 1532
"SPIDER-MAN/PETER PAR": 1745
"CAPTAIN AMERICA": 1937
```

Figure 2: Outputs for all tasks in the assignment 3.

4 Spark SQL (Assigment 4)

4.1 Task 1

- Create a SparkSession.
- Define a schema for the .csv file.
- Read the CSV file into a DataFrame with the specified schema.
- Register the DataFrame as a temporary SQL table named ”temps”.
- Execute a SQL query to find the minimum temperature (TMIN) for each stationID.
- Collect the query results into a list of rows.
- Iterate over the results.

4.2 Tasks 2 and 3

- Create a SparkSession.
- Read the contents of the file Book.

- Split each line into words, remove empty strings, and create a new column word.
- Register the word_df DataFrame as a temporary SQL table named "words".
- Execute a SQL query to count occurrences of each word, group by word, and order by count in descending order; show the results.

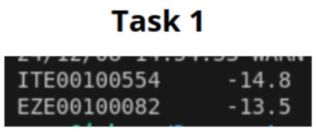
4.3 Tasks 4 and 5

- Create a Spark session.
- Read the CSV file into a DataFrame without headers.
- Assign column names to the DataFrame.
- Register the DataFrame as a temporary SQL table.
- Write an SQL query to calculate the total amount spent by each customer.
- Execute the SQL query and show results

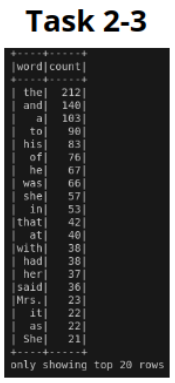
4.4 Tasks 6 and 7

- Create a Spark session.
- Load the graph dataset as an RDD and count superhero occurrences.
- Convert the graph RDD to a DataFrame with column names.
- Define a schema for the names dataset.
- Load the names dataset and convert it to a DataFrame.
- Register both DataFrames as temporary SQL tables.
- Write an SQL query to join the datasets and sort by frequency.
- Execute the SQL query and show the results.

4.5 Terminal Outputs

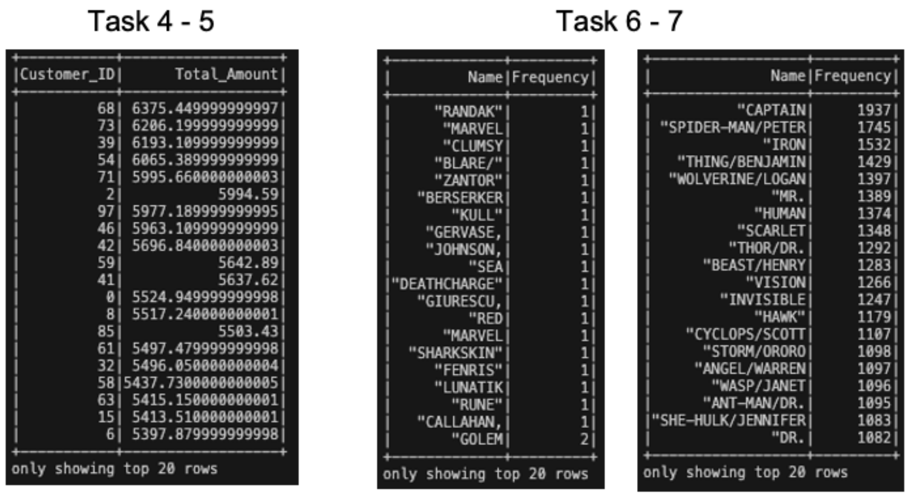


```
ITE00100554      -14.8
EZE00100082      -13.5
```



```
+----+-----+
|word|count|
+----+-----+
| the|  212|
| and|  140|
|   a|  103|
|  to|   90|
| his|   83|
|  of|   76|
|  he|   67|
| was|   66|
| she|   57|
|  in|   53|
|that|   42|
|  at|   40|
|with|   38|
| had|   38|
| her|   37|
|said|   36|
|Mrs.|   23|
|  it|   22|
|  as|   22|
| She|   21|
+----+-----+
only showing top 20 rows
```

Figure 3: Outputs for tasks 1 to 3 in the assignment 4.



```
+-----------+-----------------+
|Customer_ID|     Total_Amount|
+-----------+-----------------+
|         68| 6375.449999999997|
|         73| 6206.199999999999|
|         39| 6193.109999999999|
|         54| 6065.389999999999|
|         71| 5995.660000000003|
|          2|          5994.59|
|         97| 5977.189999999995|
|         46| 5963.109999999999|
|         42| 5696.840000000003|
|         59|          5642.89|
|         41|          5637.62|
|          0| 5524.949999999998|
|          8| 5517.240000000001|
|         85|          5503.43|
|         61| 5497.479999999998|
|         32| 5496.050000000004|
|         58|5437.730000000005|
|         63| 5415.150000000001|
|         15| 5413.510000000001|
|          6| 5397.879999999998|
+-----------+-----------------+
only showing top 20 rows
```

```
+--------------+---------+
|          Name|Frequency|
+--------------+---------+
|      "RANDAK"|        1|
|       "MARVEL|        1|
|       "CLUMSY|        1|
|      "BLARE/"|        1|
|      "ZANTOR"|        1|
|    "BERSERKER|        1|
|        "KULL"|        1|
|     "GERVASE,|        1|
|     "JOHNSON,|        1|
|          "SEA|        1|
| "DEATHCHARGE"|        1|
|    "GIURESCU,|        1|
|          "RED|        1|
|       "MARVEL|        1|
|   "SHARKSKIN"|        1|
|      "FENRIS"|        1|
|      "LUNATIK|        1|
|        "RUNE"|        1|
|    "CALLAHAN,|        1|
|        "GOLEM|        2|
+--------------+---------+
only showing top 20 rows
```

```
+------------------+---------+
|              Name|Frequency|
+------------------+---------+
|          "CAPTAIN|     1937|
| "SPIDER-MAN/PETER|     1745|
|             "IRON|     1532|
|   "THING/BENJAMIN|     1429|
|  "WOLVERINE/LOGAN|     1397|
|              "MR.|     1389|
|            "HUMAN|     1374|
|          "SCARLET|     1348|
|         "THOR/DR.|     1292|
|      "BEAST/HENRY|     1283|
|           "VISION|     1266|
|        "INVISIBLE|     1247|
|            "HAWK"|     1179|
|    "CYCLOPS/SCOTT|     1107|
|      "STORM/ORORO|     1098|
|     "ANGEL/WARREN|     1097|
|       "WASP/JANET|     1096|
|      "ANT-MAN/DR.|     1095|
|"SHE-HULK/JENNIFER|     1083|
|              "DR.|     1082|
+------------------+---------+
only showing top 20 rows
```

Figure 4: Outputs for tasks 4 to 7 in the assignment 4.