Recommendation Engine

September 23, 2024

Importing Libraries

[1]:
```
import pyspark
import glob
import os
import pandas as pd
from pyspark.sql import SparkSession
from pyspark.sql.functions import lower, when, col, count, desc
from pyspark.sql.functions import concat_ws, lit, expr
from pyspark.sql import Window
from pyspark.sql.functions import rank
!pip install Pyppeteer
!pyppteer-install
!pip install fpdf
from fpdf import FPDF
import inspect
```

```
Requirement already satisfied: Pyppeteer in
/opt/anaconda3/envs/pyspark_env/lib/python3.8/site-packages (2.0.0)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in
/opt/anaconda3/envs/pyspark_env/lib/python3.8/site-packages (from Pyppeteer)
(1.4.4)
Requirement already satisfied: certifi>=2023 in
/opt/anaconda3/envs/pyspark_env/lib/python3.8/site-packages (from Pyppeteer)
(2024.8.30)
Requirement already satisfied: importlib-metadata>=1.4 in
/opt/anaconda3/envs/pyspark_env/lib/python3.8/site-packages (from Pyppeteer)
(7.0.1)
Requirement already satisfied: pyee<12.0.0,>=11.0.0 in
/opt/anaconda3/envs/pyspark_env/lib/python3.8/site-packages (from Pyppeteer)
(11.1.1)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in
/opt/anaconda3/envs/pyspark_env/lib/python3.8/site-packages (from Pyppeteer)
(4.66.5)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in
/opt/anaconda3/envs/pyspark_env/lib/python3.8/site-packages (from Pyppeteer)
(1.26.20)
Requirement already satisfied: websockets<11.0,>=10.0 in
/opt/anaconda3/envs/pyspark_env/lib/python3.8/site-packages (from Pyppeteer)
```

```
(10.4)
Requirement already satisfied: zipp>=0.5 in
/opt/anaconda3/envs/pyspark_env/lib/python3.8/site-packages (from importlib-
metadata>=1.4->Pyppeteer) (3.17.0)
Requirement already satisfied: typing-extensions in
/opt/anaconda3/envs/pyspark_env/lib/python3.8/site-packages (from
pyee<12.0.0,>=11.0.0->Pyppeteer) (4.11.0)
zsh:1: command not found: pyppteer-install
Requirement already satisfied: fpdf in
/opt/anaconda3/envs/pyspark_env/lib/python3.8/site-packages (1.7.2)
```

Initialize the Spark Session

[2]:
```
spark = SparkSession.builder.master("local[1]") \
                    .appName('Recommendation Engine') \
                    .getOrCreate()
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
24/09/16 23:10:58 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
24/09/16 23:10:58 WARN Utils: Service 'SparkUI' could not bind on port 4040.
Attempting port 4041.
24/09/16 23:10:58 WARN Utils: Service 'SparkUI' could not bind on port 4041.
Attempting port 4042.
```

Define Function

Function1: unique_dataset

[3]:
```
def unique_dataset(df):
    """
    The unique_dataset function removes duplicates based on the SKU number,␣
 ↪keeping the first occurrence.
    @param df: The original DataFrame that contains all the SKUs.
    @return: df_unique: A DataFrame that contains only the unique SKUs.
    """
    # Drop duplicates based on the 'sku' column, keeping the first occurrence
    df_unique = df.dropDuplicates(['sku'])

    # Return the updated DataFrame
    return df_unique
```

```
24/09/16 23:11:14 WARN GarbageCollectionMetrics: To enable non-built-in garbage
collector(s) List(G1 Concurrent GC), users should configure it(them) to
spark.eventLog.gcMetrics.youngGenerationGarbageCollectors or
spark.eventLog.gcMetrics.oldGenerationGarbageCollectors
```

Function2: check_sku

[4]:
```
def check_sku(df_unique, sku):
    '''
    The check_sku function checks whether a target SKU is present in the dataset.
    @params: df_unique: the dataframe that contains only the unique SKUs; sku:␣
↪the SKU entered by the user.
    @return: a message showing whether the sku is present in the data.
    '''
    sku_df = df_unique.filter(lower(df_unique['sku']) == sku.lower())
    if sku_df.count() > 0:
        print(f"{sku} is present in our dataset.")
        # Display the DataFrame result
        df_result = sku_df.collect()  # You can use .show() for direct display,␣
↪or .collect() for processing
    else:
        print(f"{sku} is not present in our DataFrame.")
```

Function3: transposeRDD

[5]:
```
def transposeRDD(df_unique,sku):
    '''
    The transposeRDD function transposes the dataframe and turn the dataframe␣
↪into an RDD.
    @params: df_unique: the dataframe that contains all the unique SKUs; sku:␣
↪the SKU entered by the user.
    @return: transpose_RDD: an RDD with each attributes as a column.
    '''
    transpose_RDD = df_unique.rdd.map(lambda x: (x["sku"], x["attributes"][0],␣
↪x["attributes"][1], x["attributes"][2], x["attributes"][3],␣
↪x["attributes"][4], x["attributes"][5], x["attributes"][6],␣
↪x["attributes"][7], x["attributes"][8], x["attributes"][9]))
    return transpose_RDD
```

Function4: filterSku

[6]:
```
def filterSku(transpose_RDD, sku):
    '''
    The filterSKU function filters out the target SKU.
    @params: transpose_RDD: an RDD with each attributes as a column; sku: the␣
↪SKU entered by the user.
    @return: df_sku_view: a view of the data frame without the target SKU for␣
↪comparison; target_row: the target SKU and its attributes.
    '''
    target_rdd = transpose_RDD.filter(lambda row: row[0] == sku)
    filtered_rdd = transpose_RDD.filter(lambda row: row[0] != sku)

    columns = ["Sku", "Att_a", "Att_b", "Att_c", "Att_d", "Att_e", "Att_f",␣
↪"Att_g", "Att_h", "Att_i", "Att_j"]
```

```
    # Convert the filtered_rdd to a DataFrame with column names
    filtered_df = filtered_rdd.toDF(columns)
    filtered_df.createOrReplaceTempView("sku_view")

    # Assuming rdd_result is an RDD containing a single row (target SKU row)
    target_row = target_rdd.first()  # Extracts the tuple

    # Load the sku_view into a DataFrame
    df_sku_view = spark.sql("SELECT * FROM sku_view")

    return df_sku_view, target_row
```

Function5: SkuWeight

[7]:
```
def SkuWeight(df_sku_view,target_row):
    '''
    The SkuWeight function calculates the similarity that every SKU to the␣
↪target SKU.
    @params: df_sku_view: a view of the data frame without the target SKU for␣
↪comparison; target_row: the target SKU and its attributes.
    @return: df_with_weight: a data frame that has a column of the similarity␣
↪that  every SKU to the target SKU.
    '''
     # Perform the comparison and create the new columns using direct indexing
    df_with_scores = df_sku_view.withColumn("Va", when(col("Att_a") ==␣
↪target_row[1], 9).otherwise(-1)) \
    .withColumn("Vb", when(col("Att_b") == target_row[2], 8).otherwise(-1)) \
    .withColumn("Vc", when(col("Att_c") == target_row[3], 7).otherwise(-1)) \
    .withColumn("Vd", when(col("Att_d") == target_row[4], 6).otherwise(-1)) \
    .withColumn("Ve", when(col("Att_e") == target_row[5], 5).otherwise(-1)) \
    .withColumn("Vf", when(col("Att_f") == target_row[6], 4).otherwise(-1)) \
    .withColumn("Vg", when(col("Att_g") == target_row[7], 3).otherwise(-1)) \
    .withColumn("Vh", when(col("Att_h") == target_row[8], 2).otherwise(-1)) \
    .withColumn("Vi", when(col("Att_i") == target_row[9], 1).otherwise(-1)) \
    .withColumn("Vj", when(col("Att_j") == target_row[10], 0).otherwise(-1))

    # Register the transformed DataFrame as a new view
    df_with_scores.createOrReplaceTempView("sku_comparison_view")

    # Create a new column "weight" by concatenating non-"-1" values and casting␣
↪to integer
    df_with_weight = df_with_scores.withColumn(
        "Weight",
        expr("CAST(concat_ws('', " + ", ".join([f"CASE WHEN {col} != -1 THEN␣
↪CAST({col} AS STRING) ELSE '' END" for col in ['Va', 'Vb', 'Vc', 'Vd', 'Ve',␣
↪'Vf', 'Vg', 'Vh', 'Vi', 'Vj']]) + ") AS INT)")
```

```
    )
    return df_with_weight
```

Function6: GetSimilarWeightSku

[8]:
```
def GetSimilarWeightSku(df_with_weight, sku):
    '''
    The GetSimilarWeightSku function oders the SKUs by their similarity to the␣
 ↪target SKU.
    @params: df_with_weight: a data frame that has a column of the similarity␣
 ↪that  every SKU to the target SKU; sku: the SKU entered by the user.
    @return: df_with_rank: a data frame that has a column of the rank of the␣
 ↪similarity that  every SKU to the target SKU.
    '''
    # Define a window specification to rank by weight in descending order
    window_spec = Window.orderBy(col("weight").desc())

    # Assign a rank to every SKU by weight in descending order
    df_with_rank = df_with_weight.withColumn("rank", rank().over(window_spec))

    return df_with_rank
```

Function7: TopSimilarSku

[9]:
```
def TopSimilarSku(df_with_rank, sku):
    '''
    The TopSimilarSku shows the top 10 similar SKUs to the target SKU.
    @params: df_with_rank: a data frame that has a column of the rank of the␣
 ↪similarity that  every SKU to the target SKU; sku: the SKU entered by the user.
    @return: top_10_similar_skus: a data frame that shows the top 10 similar␣
 ↪SKUs to the target SKU.
    '''
    # Order by rank in ascending order to get the top ranks
    df_with_rank_ordered = df_with_rank.orderBy("rank")

    # Get the top 10 rows based on the rank
    df_with_rank_ordered_10 = df_with_rank_ordered.limit(10)

    #keep only sku and attributes
    #concatenate all attribute columns into a single string column
    attribute_columns = ["Att_a", "Att_b", "Att_c", "Att_d", "Att_e", "Att_f",␣
 ↪"Att_g", "Att_h", "Att_i", "Att_j"]

    # Combine all attributes into a single 'attribute' column
    df_consolidated = df_with_rank_ordered_10.withColumn(
        "Attribute",
        concat_ws(", ", *[col(attr) for attr in attribute_columns])
    )
```

```
    # Select only the sku and the new attribute column
    top_10_similar_skus = df_consolidated.select("Sku", "Weight", "Attribute")

    return top_10_similar_skus
```

Main Function

[10]:
```
if __name__ == "__main__":
    file_path = glob.glob('**/dataset.json')
    file_path = os.path.abspath(file_path[0])
    df = spark.read.json(file_path)

    target_sku = 'sku-100'

    # function1: Run the unique_dataset function to remove duplicates
    df_unique = unique_dataset(df)

    # function2: Check if a specific SKU is present in the updated DataFrame
    check_sku(df_unique, target_sku)

    # function3: transpose df into rdd with transposeRDD function
    transpose_RDD = transposeRDD(df_unique, target_sku)

    # function4: filter out the target sku
    filterSku(transpose_RDD, target_sku)
    df_sku_view, target_row = filterSku(transpose_RDD, target_sku)

    # function5: calculate the weight
    df_with_weight = SkuWeight(df_sku_view,target_row)

    # function6: calculate similar weight
    df_with_rank = GetSimilarWeightSku(df_with_weight,target_sku)

    # function7: get top 10 similar SKUs using TopSimilarSku
    top_10_similar_skus = TopSimilarSku(df_with_rank, target_sku)

    # return output: display the top 10 similar SKUs
    top_10_similar_skus.show(truncate=False)
```

```
sku-100 is present in our dataset.

24/09/16 23:12:29 WARN WindowExec: No Partition Defined for Window operation!
Moving all data to a single partition, this can cause serious performance
degradation.
24/09/16 23:12:29 WARN WindowExec: No Partition Defined for Window operation!
Moving all data to a single partition, this can cause serious performance
degradation.
24/09/16 23:12:29 WARN WindowExec: No Partition Defined for Window operation!
```

```
Moving all data to a single partition, this can cause serious performance
degradation.
24/09/16 23:12:30 WARN WindowExec: No Partition Defined for Window operation!
Moving all data to a single partition, this can cause serious performance
degradation.
24/09/16 23:12:30 WARN WindowExec: No Partition Defined for Window operation!
Moving all data to a single partition, this can cause serious performance
degradation.

+---------+------+-------------------------------------------------------------
------------------------------+
|Sku      |Weight|Attribute
|
+---------+------+-------------------------------------------------------------
------------------------------+
|sku-9124 |98652 |att-a-13, att-b-5, att-c-9, att-d-4, att-e-5, att-f-8,
att-g-5, att-h-7, att-i-3, att-j-13   |
|sku-13907|87643 |att-a-11, att-b-5, att-c-4, att-d-4, att-e-12, att-f-7,
att-g-12, att-h-11, att-i-12, att-j-6|
|sku-1884 |86543 |att-a-8, att-b-5, att-c-12, att-d-4, att-e-5, att-f-7,
att-g-12, att-h-13, att-i-6, att-j-4  |
|sku-9347 |86432 |att-a-5, att-b-5, att-c-7, att-d-4, att-e-9, att-f-7,
att-g-12, att-h-7, att-i-15, att-j-9   |
|sku-13127|9864  |att-a-13, att-b-5, att-c-3, att-d-4, att-e-8, att-f-7,
att-g-13, att-h-5, att-i-3, att-j-8   |
|sku-11408|9862  |att-a-13, att-b-5, att-c-9, att-d-4, att-e-13, att-f-14,
att-g-10, att-h-7, att-i-4, att-j-7 |
|sku-3211 |9851  |att-a-13, att-b-5, att-c-13, att-d-1, att-e-5, att-f-8,
att-g-10, att-h-8, att-i-9, att-j-9  |
|sku-14868|9843  |att-a-13, att-b-5, att-c-15, att-d-13, att-e-9, att-f-7,
att-g-12, att-h-10, att-i-1, att-j-5|
|sku-18342|9840  |att-a-13, att-b-5, att-c-10, att-d-5, att-e-7, att-f-7,
att-g-15, att-h-13, att-i-14, att-j-3|
|sku-5077 |9830  |att-a-13, att-b-5, att-c-2, att-d-7, att-e-4, att-f-8,
att-g-12, att-h-4, att-i-4, att-j-3   |
+---------+------+-------------------------------------------------------------
------------------------------+
```