sentimentvisualizer

August 6, 2024

[1]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
import pandas as pd
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output, State
import plotly.graph_objs as go
from datetime import datetime,timedelta
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
  """
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
```

[2]:
```
#Spark Session creation configured to interact with MongoDB
spark = SparkSession.builder.appName("pyspark-notebook").\
config("spark.jars.packages","org.apache.spark:spark-sql-kafka-0-10_2.12:3.0.
 ↪0,org.apache.spark:spark-avro_2.12:3.0.0,org.mongodb.spark:
 ↪mongo-spark-connector_2.12:3.0.0").\
config("spark.mongodb.input.uri","mongodb://ubuntu_mongo_1:27017/twitter_db.
 ↪tweets").\
config("spark.mongodb.output.uri","mongodb://ubuntu_mongo_1:27017/twitter_db.
 ↪tweets").\
getOrCreate()
```

```
Ivy Default Cache set to: /root/.ivy2/cache
The jars for the packages stored in: /root/.ivy2/jars
:: loading settings :: url = jar:file:/usr/local/lib/python3.7/dist-packages/pys
park/jars/ivy-2.4.0.jar!/org/apache/ivy/core/settings/ivysettings.xml
org.apache.spark#spark-sql-kafka-0-10_2.12 added as a dependency
org.apache.spark#spark-avro_2.12 added as a dependency
org.mongodb.spark#mongo-spark-connector_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-
```

```
parent-d46168af-7c67-4692-8e92-3152ef39a1e0;1.0
        confs: [default]
        found org.apache.spark#spark-sql-kafka-0-10_2.12;3.0.0 in central
        found org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.0.0 in
central
        found org.apache.kafka#kafka-clients;2.4.1 in central
        found com.github.luben#zstd-jni;1.4.4-3 in central
        found org.lz4#lz4-java;1.7.1 in central
        found org.xerial.snappy#snappy-java;1.1.7.5 in central
        found org.slf4j#slf4j-api;1.7.30 in central
        found org.spark-project.spark#unused;1.0.0 in central
        found org.apache.commons#commons-pool2;2.6.2 in central
        found org.apache.spark#spark-avro_2.12;3.0.0 in central
        found org.mongodb.spark#mongo-spark-connector_2.12;3.0.0 in central
        found org.mongodb#mongodb-driver-sync;4.0.5 in central
        found org.mongodb#bson;4.0.5 in central
        found org.mongodb#mongodb-driver-core;4.0.5 in central
:: resolution report :: resolve 494ms :: artifacts dl 8ms
        :: modules in use:
        com.github.luben#zstd-jni;1.4.4-3 from central in [default]
        org.apache.commons#commons-pool2;2.6.2 from central in [default]
        org.apache.kafka#kafka-clients;2.4.1 from central in [default]
        org.apache.spark#spark-avro_2.12;3.0.0 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.12;3.0.0 from central in
[default]
        org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.0.0 from central
in [default]
        org.lz4#lz4-java;1.7.1 from central in [default]
        org.mongodb#bson;4.0.5 from central in [default]
        org.mongodb#mongodb-driver-core;4.0.5 from central in [default]
        org.mongodb#mongodb-driver-sync;4.0.5 from central in [default]
        org.mongodb.spark#mongo-spark-connector_2.12;3.0.0 from central in
[default]
        org.slf4j#slf4j-api;1.7.30 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.7.5 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   14  |   0   |   0   |   0   ||   14  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-
parent-d46168af-7c67-4692-8e92-3152ef39a1e0
        confs: [default]
        0 artifacts copied, 14 already retrieved (0kB/9ms)
24/07/22 06:44:00 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
```

```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
24/07/22 06:44:02 WARN Utils: Service 'SparkUI' could not bind on port 4040.
Attempting port 4041.
24/07/22 06:44:02 WARN Utils: Service 'SparkUI' could not bind on port 4041.
Attempting port 4042.
24/07/22 06:44:02 WARN Utils: Service 'SparkUI' could not bind on port 4042.
Attempting port 4043.
```

[3]:
```
app = dash.Dash(__name__)

#Color assignment
colors = {
    'background': 'white',#'#0C0F0A',
    'text': '#FFFFFF'
}

def create_header(title):
    """Takes the input and Returns a html header

    Parameters
    ----------
    title : String
        Title of the Dashboard

    Returns
    ----------
        header: html header
    """

    header_style = {
        'background-color' : '#1B95E0',
        'padding' : '1.5rem',
        'color': 'white',
        'font-family': 'Verdana, Geneva, sans-serif'
    }
    header = html.Header(html.H1(children=title, style=header_style))
    return header

def generate_table(df, max_rows=10):
    """Takes pandas dataframe, optional max number of rows to display and␣
 ↪returns html table

    Parameters
    ----------
    df : DataFrame
        Pandas dataframe
    max_rows: int
        Number of max rows to fit in a table

    Returns
    ----------
        table: html table
    """

    table = html.Table(className="responsive-table",
                      children=[
                          html.Thead(
                              html.Tr(
                                  children=[html.Th(col.title()) for col in df.
 ↪columns.values]
                                  )
                              ),
                          html.Tbody(
                              [
                              html.Tr(
                                  children=[html.Td(data) for data in d]
                                  )
                               for d in df.values.tolist()])
                          ]
    )

    return table

#Layout definition - contains a header, input box to get search term, a graph␣
 ↪and a table
app.layout = html.Div(style={'backgroundColor': colors['background']}, children=
    [
        html.Div([create_header('Live Dashboard - Twitter Sentiment␣
 ↪Analysis')]),
        html.Div(["Serch Term: ", dcc.Input(id='sentiment_term',␣
 ↪value='twitter', type='text',placeholder='Enter word to be searched'),
                  dcc.Graph(id='live-graph', animate=False)
                 ]
                 ,style={'width': '64%', 'display': 'inline-block'}
                ),
        html.Div([html.H2("Recent Tweets"),
                  html.Div(id="recent-tweets-table")]
                 ,style={'width': '34%', 'display': 'inline-block'}
                ),
        #Intervals define the frequency in which the html element should be␣
 ↪updated
```

```
        dcc.Interval(id='graph-update',interval=1*1000, n_intervals=0),
        dcc.Interval(id='recent-table-update',interval=10*1000, n_intervals=0)
    ]
)
```

[ ]:
```
#Call back for live graph
@app.callback(Output('live-graph', 'figure'),
              Input('graph-update', 'n_intervals'),
              Input('sentiment_term', 'value')
             )
def update_graph_scatter(n_intervals,sentiment_term):
    """Takes interval and search term as inputs and returs live-graph

    Parameters
    ----------
    n_intervals : int
        Frequency to update figure
    sentiment_term: int
        Search term to analyse the sentiment

    Returns
    ----------
        graph: html graph
        live-graph
    """
    try:
        #Read data from MongoDB for last 200 seconds
        time_diff = (datetime.utcnow() - timedelta(seconds=200)).
 ↪strftime('%Y-%m-%d %H:%M:%S')
        df = spark.read.format("mongo").load().
 ↪select("timestamp_ms","text","prediction").
 ↪where("timestamp_ms>'"+time_diff+"' and lower(text) like␣
 ↪lower('%"+sentiment_term+"%')").toPandas()
        df.sort_values('timestamp_ms', inplace=True)
        df.dropna(inplace=True)

        #Define X and Y axis values
        X = df["timestamp_ms"]
        Y = df['prediction']#[-100:]

        #Scatter graph definition
        data = go.Scatter(
                x=X,
                y=Y,
                name='Scatter',
                mode= 'lines+markers'
                )
        return {'data': [data],'layout' : go.Layout(xaxis=dict(range=[X.min(),X.
↪max()]),
                                                    yaxis=dict(range=[0,1]),
                                                    title='Twitter Sentiment␣
↪{}'.format(sentiment_term)
                                                   )
               }

    except Exception as e:
        #File to capture exceptions
        with open('errors.txt','a') as f:
            f.write(str(e))
            f.write('\n')

#Call back for table to populate latest 10 tweets
@app.callback(Output('recent-tweets-table', 'children'),
              Input('recent-table-update', 'n_intervals'),
              Input('sentiment_term', 'value')
             )
def update_recent_tweets(n_intervals,sentiment_term):
    """Takes interval and search term as inputs and returs live-graph

    Parameters
    ----------
    n_intervals : int
        Frequency to update figure
    sentiment_term: int
        Search term to analyse the sentiment

    Returns
    ----------
        table: html graph
        table of latest 10 tweets
    """

    try:
        #Read data from MongoDB for last 200 seconds
        time_diff = (datetime.utcnow() - timedelta(seconds=200)).
↪strftime('%Y-%m-%d %H:%M:%S')
        df = spark.read.format("mongo").load().
↪select("timestamp_ms","text","prediction").
↪where("timestamp_ms>'"+time_diff+"' and lower(text) like␣
↪lower('%"+sentiment_term+"%')").limit(5).toPandas()
        df['sentiment'] = df['prediction']
        df['timestamp'] = df['timestamp_ms']
        df['tweet']     = df['text']
        df.drop(['timestamp_ms','text'],axis=1)

        df = df[['timestamp','tweet','sentiment']]

        return generate_table(df, max_rows=5)
    except Exception as e:
        #File to capture exceptions
        with open('table_errors.txt','a') as f:
            f.write(str(e))
            f.write('\n')

if __name__ == '__main__':
    app.run_server(debug=False, use_reloader=False, port=8050,host= '0.0.0.0')
```

```
Dash is running on http://0.0.0.0:8050/

 * Serving Flask app '__main__' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: off

 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production
deployment.
 * Running on http://172.22.0.4:8050/ (Press CTRL+C to quit)
113.172.146.187 - - [22/Jul/2024 06:46:03] "GET / HTTP/1.1" 200 -
113.172.146.187 - - [22/Jul/2024 06:46:03] "GET /_dash-layout HTTP/1.1" 200 -
113.172.146.187 - - [22/Jul/2024 06:46:04] "GET /_dash-dependencies HTTP/1.1"
200 -
113.172.146.187 - - [22/Jul/2024 06:46:04] "GET /_favicon.ico?v=2.0.0 HTTP/1.1"
200 -
113.172.146.187 - - [22/Jul/2024 06:46:04] "GET /_dash-component-
suites/dash/dcc/async-graph.js HTTP/1.1" 200 -
113.172.146.187 - - [22/Jul/2024 06:46:04] "GET /_dash-component-
suites/dash/dcc/async-plotlyjs.js HTTP/1.1" 200 -
```

[ ]:

```
```