assignment03_Muley_Tushar

January 16, 2022

Name: Tushar Muley

Assignment: Week 03 - Assignment 03

Date:January 16, 2022

Assignment 3

[2]:
```
import os
import sys
import gzip
import json
from pathlib import Path
import csv

import pandas as pd
import s3fs
import pyarrow as pa
from pyarrow.json import read_json
import pyarrow.parquet as pq
import fastavro
import pygeohash
import snappy
import jsonschema
from jsonschema.exceptions import ValidationError
```

[37]:
```
# simplifed by loading file directly to location due to error about SSL␣
↪certificates
src_data_path = 'routes.jsonl.gz'
results_dir = current_dir.joinpath('results')
current_dir = Path(os.getcwd()).absolute()


def read_jsonl_data():
    with gzip.open(src_data_path, 'rb') as f:
        records = [json.loads(line) for line in f.readlines()]
        return records
```

[38]:
```
# load records
records = read_jsonl_data()
```

[39]:
```
# print recoreds for viewing
print(records[0])
```

```
{'airline': {'airline_id': 410, 'name': 'Aerocondor', 'alias': 'ANA All Nippon
Airways', 'iata': '2B', 'icao': 'ARD', 'callsign': 'AEROCONDOR', 'country':
'Portugal', 'active': True}, 'src_airport': {'airport_id': 2965, 'name': 'Sochi
International Airport', 'city': 'Sochi', 'country': 'Russia', 'iata': 'AER',
'icao': 'URSS', 'latitude': 43.449902, 'longitude': 39.9566, 'altitude': 89,
'timezone': 3.0, 'dst': 'N', 'tz_id': 'Europe/Moscow', 'type': 'airport',
'source': 'OurAirports'}, 'dst_airport': {'airport_id': 2990, 'name': 'Kazan
International Airport', 'city': 'Kazan', 'country': 'Russia', 'iata': 'KZN',
'icao': 'UWKD', 'latitude': 55.606201171875, 'longitude': 49.278701782227,
'altitude': 411, 'timezone': 3.0, 'dst': 'N', 'tz_id': 'Europe/Moscow', 'type':
'airport', 'source': 'OurAirports'}, 'codeshare': False, 'equipment': ['CR2']}
```

3.1.a JSON Schema

[40]:
```
def validate_jsonl_data(records):
    schema_path = schema_dir.joinpath('routes-schema.json')
    validation_csv_path = results_dir.joinpath('validation-results.csv')
    with open(schema_path) as f:
        schema = json.load(f)

    with open(validation_csv_path, 'w') as f:
        for i, record in enumerate(records):
            try:
                ## TODO: Validate record
                jsonschema.validate(instance=record, schema=schema)
                pass
            except ValidationError as e:
                ## Print message if invalid record
                print('Exception while reading record:', i)
                f.write(str(e))
                pass
```

[41]:
```
validate_jsonl_data(records)
```

3.1.b Avro

[42]:
```
def create_avro_dataset(records):
    schema_path = schema_dir.joinpath('routes.avsc')
    data_path = results_dir.joinpath('routes.avro')
    ## TODO: Use fastavro to create Avro dataset
    with open(schema_path) as f:
        schema = json.loads(f.read())

    parsed_schema = fastavro.parse_schema(schema)
    # writing
    with open(data_path, 'wb') as out:
```

```
        fastavro.writer(out, parsed_schema, records)

    # reading record
    with open(data_path, 'rb') as fo:
        for record in fastavro.reader(fo):
            print(record)
            break


create_avro_dataset(records)
```

```
{'airline': {'airline_id': 410, 'name': 'Aerocondor', 'alias': 'ANA All Nippon
Airways', 'iata': '2B', 'icao': 'ARD', 'callsign': 'AEROCONDOR', 'country':
'Portugal', 'active': True}, 'src_airport': {'airport_id': 2965, 'name': 'Sochi
International Airport', 'city': 'Sochi', 'iata': 'AER', 'icao': 'URSS',
'latitude': 43.449902, 'longitude': 39.9566, 'timezone': 3.0, 'dst': 'N',
'tz_id': 'Europe/Moscow', 'type': 'airport', 'source': 'OurAirports'},
'dst_airport': {'airport_id': 2990, 'name': 'Kazan International Airport',
'city': 'Kazan', 'iata': 'KZN', 'icao': 'UWKD', 'latitude': 55.606201171875,
'longitude': 49.278701782227, 'timezone': 3.0, 'dst': 'N', 'tz_id':
'Europe/Moscow', 'type': 'airport', 'source': 'OurAirports'}, 'codeshare':
False, 'stops': 0, 'equipment': ['CR2']}
```

0.0.1 3.1.c Parquet

[12]:
```
# paths to all directos and files
current_dir = Path(os.getcwd()).absolute()
results_dir = current_dir.joinpath('results')



def create_parquet_dataset():
    src_data_path = 'routes.jsonl.gz'
    parquet_output_path = results_dir.joinpath('routes.parquet')
    #src_data_path = 'routes.jsonl.gz'
    #src_data_path = 'routes.jsonl'

    # not used as file is now local ***

    #s3 = s3fs.S3FileSystem(
    #    anon=True,
    #    client_kwargs={
    #        'endpoint_url': endpoint_url
    #    }
    #)
    #***
```

```
    with open(src_data_path, 'rb') as f_gz:
        with gzip.open(f_gz, 'rb') as f:
            pass
            ## TODO: Use Apache Arrow to create Parquet table and save the␣
↪dataset
            table = read_json(f)
        pq.write_table(table, parquet_output_path)

    # reading file not used
    #df = pq.read_table(source=parquet_output_path).to_pandas()
    #print(df)



create_parquet_dataset()
```

3.1.d Protocol Buffers

[50]:
```
sys.path.insert(0, os.path.abspath('routes_pb2'))

import routes_pb2

def _airport_to_proto_obj(airport):
    obj = routes_pb2.Airport()
    if airport is None:
        return None
    if airport.get('airport_id') is None:
        return None

    obj.airport_id = airport.get('airport_id')
    if airport.get('name'):
        obj.name = airport.get('name')
    if airport.get('city'):
        obj.city = airport.get('city')
    if airport.get('iata'):
        obj.iata = airport.get('iata')
    if airport.get('icao'):
        obj.icao = airport.get('icao')
    if airport.get('altitude'):
        obj.altitude = airport.get('altitude')
    if airport.get('timezone'):
        obj.timezone = airport.get('timezone')
    if airport.get('dst'):
        obj.dst = airport.get('dst')
    if airport.get('tz_id'):
        obj.tz_id = airport.get('tz_id')
    if airport.get('type'):
        obj.type = airport.get('type')
```

```
    if airport.get('source'):
        obj.source = airport.get('source')

    obj.latitude = airport.get('latitude')
    obj.longitude = airport.get('longitude')

    return obj


def _airline_to_proto_obj(airline):
    obj = routes_pb2.Airline()
    ## TODO: Create an Airline obj using Protocol Buffers API
    if airline is None:
        return None
    if airline.get('airline_id') is None:
        return None

    obj.airline_id = airline.get('airline_id')

    if airline.get('name'):
        obj.name = airline.get('name')

    if airline.get('alias'):
        obj.alias = airline.get('alias')

    if airline.get('iata'):
        obj.iata = airline.get('iata')

    if airline.get('icao'):
        obj.icao = airline.get('icao')

    if airline.get('callsign'):
        obj.callsign = airline.get('callsign')

    if airline.get('country'):
        obj.country = airline.get('country')

    obj.active = airline.get('active')
    return obj


def create_protobuf_dataset(records):
    routes = routes_pb2.Routes()
    for record in records:
        route = routes_pb2.Route()
        ## TODO: Implement the code to create the Protocol Buffers Dataset
        if route is None:
```

```
            return None

        airline = _airline_to_proto_obj(record.get('airline', {}))
        if airline:
            route.airline.CopyFrom(airline)

        if _airport_to_proto_obj(record.get('src_airport', {})) is not None:
            src_airport = _airport_to_proto_obj(record.get('src_airport', {}))
            route.src_airport.CopyFrom(src_airport)
        else:
            pass

        if _airport_to_proto_obj(record.get('dst_airport', {})) is not None:
            dst_airport = _airport_to_proto_obj(record.get('dst_airport', {}))
            route.dst_airport.CopyFrom(dst_airport)
        else:
            pass

        if record.get('codeshare'):
            route.codeshare = record.get('codeshare')
        else:
            route.codeshare = False

        if record.get('stops'):
            route.stops = record.get('stops')

        equipment = record.get('equipment')
        if len(equipment) > 1:
            for i, v in enumerate(equipment):
                route.equipment.append(v)
        else:
            equipment = record.get('equipment')

        routes.route.append(route)

    data_path = results_dir.joinpath('routes.pb')

    with open(data_path, 'wb') as f:
        f.write(routes.SerializeToString())

    compressed_path = results_dir.joinpath('routes.pb.snappy')

    with open(compressed_path, 'wb') as f:
        f.write(snappy.compress(routes.SerializeToString()))

create_protobuf_dataset(records)
```

3.2.a Simple Geohash Index

[51]:
```
def create_hash_dirs(records):
    geoindex_dir = results_dir.joinpath('geoindex')
    geoindex_dir.mkdir(exist_ok=True, parents=True)
    hashes = []
    ## TODO: Create hash index
    for record in records:
        src_airport = record.get('src_airport', {})
        if src_airport:
            latitude = src_airport.get('latitude')
            longitude = src_airport.get('longitude')
            if latitude and longitude:
                hashes.append(pygeohash.encode(latitude, longitude))

    hashes.sort()

    three_letter = sorted(list(set([entry[:3] for entry in hashes])))

    hash_index = {value: [] for value in three_letter}

    for record in records:
        geohash = record.get('geohash')
        if geohash:
            hash_index[geohash[:3]].append(record)

    for key, values in hash_index.items():
        output_dir = geoindex_dir.joinpath(str(key[:1])).joinpath(str(key[:2]))
        output_dir.mkdir(exist_ok=True, parents=True)
        output_path = output_dir.joinpath('{}.jsonl.gz'.format(key))
        with gzip.open(output_path, 'w') as f:
            json_output = '\n'.join([json.dumps(value) for value in values])
            f.write(json_output.encode('utf-8'))

create_hash_dirs(records)
```

3.2.b Simple Search Feature

[52]:
```
def airport_search(latitude, longitude):
    ## TODO: Create simple search to return nearest airport
    pass
    h = pygeohash.encode(latitude,longitude)
    distance = 0
    name = ''
    for i,record in enumerate(records):
        src_airport = record.get('src_airport', {})
        if src_airport:
            latitude = src_airport.get('latitude')
            longitude = src_airport.get('longitude')
```

```
            airport_name = src_airport.get('name')
            if latitude and longitude:
                h1 = pygeohash.encode(latitude,longitude)
                distance_n = pygeohash.geohash_approximate_distance(h,h1)
                if i==0:
                    distance = distance_n
                else:
                    if distance > distance_n:
                        distance = distance_n
                        name = airport_name

    print(name)

airport_search(41.1499988, -95.91779)
```

```
Eppley Airfield
```

[ ]: