Assignment 3

December 21, 2020

1 Assignment 3

Gourav Verma DSC 650 Winter 2020

Import libraries and define common helper functions

[1]:

```
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


endpoint_url='https://storage.budsc.midwest-datascience.com'

current_dir = Path(os.getcwd()).absolute()
schema_dir = current_dir.joinpath('schemas')
results_dir = current_dir.joinpath('results')
results_dir.mkdir(parents=True, exist_ok=True)


def read_jsonl_data():
    s3 = s3fs.S3FileSystem(
        anon=True,
        client_kwargs={
            'endpoint_url': endpoint_url
        }
    )
    src_data_path = 'data/processed/openflights/routes.jsonl.gz'
    with s3.open(src_data_path, 'rb') as f_gz:
        with gzip.open(f_gz, 'rb') as f:
            records = [json.loads(line) for line in f.readlines()]


    return records
```

Load the records from https://storage.budsc.midwest-datascience.com/data/processed/openflights/routes.jsonl.gz

[2]:

```
records = read_jsonl_data()
```

1.1 3.1

1.1.1 3.1.a JSON Schema

[3]:

```
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-', i)
                f.write(str(e))
                pass


validate_jsonl_data(records)
```

1.1.2 3.1.b Avro

[4]:

```
def create_avro_dataset(records):
    schema_path = schema_dir.joinpath('routes.avsc')
    data_path = results_dir.joinpath('routes.avro')
    ## TODO
    # Read schema
    with open(schema_path) as fo:
        schema = json.loads(fo.read())

    parsed_schema = fastavro.parse_schema(schema)

    # Write dataset
    with open(data_path, 'wb') as out:
        fastavro.writer(out, parsed_schema, records)

create_avro_dataset(records)
```

1.1.3 3.1.c Parquet

[5]:

```
def create_parquet_dataset():
    src_data_path = 'data/processed/openflights/routes.jsonl.gz'
    parquet_output_path = results_dir.joinpath('routes.parquet')
    s3 = s3fs.S3FileSystem(
        anon=True,
        client_kwargs={
            'endpoint_url': endpoint_url
        }
    )

    with s3.open(src_data_path, 'rb') as f_gz:
        with gzip.open(f_gz, 'rb') as f:
            pass
            ## TODO
            table = read_json(f)

    pq.write_table(table, parquet_output_path)

create_parquet_dataset()
```

1.1.4 3.1.d Protocol Buffers

[6]:

```
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
    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')
    if airline.get('active'):
        obj.active = airline.get('active')
    else:
        obj.active = False

    return obj


def create_protobuf_dataset(records):
    routes = routes_pb2.Routes()
    for record in records:
        route = routes_pb2.Route()
        ## TODO:
        airline = _airline_to_proto_obj(record.get('airline', {}))
        if airline:
            route.airline.CopyFrom(airline)
        src_airport = _airport_to_proto_obj(record.get('src_airport', {}))
        if src_airport:
            route.src_airport.CopyFrom(src_airport)
        dst_airport = _airport_to_proto_obj(record.get('dst_airport', {}))
        if dst_airport:
            route.dst_airport.CopyFrom(dst_airport)

        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)
```

1.2 3.2

1.2.1 3.2.a Simple Geohash Index

[7]:

```
def create_hash_dirs(records):
    geoindex_dir = results_dir.joinpath('geoindex')
    geoindex_dir.mkdir(exist_ok=True, parents=True)
    hashes = []
    ## TODO:
    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)
```

1.2.2 3.2.b Simple Search Feature

[16]:

```
def airport_search(latitude, longitude):
    ## TODO
    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:
                pygeohash.encode(latitude, longitude)

    print(latitude, longitude)

    return pygeohash.geohash_approximate_distance(latitude, longitude)

#    pass

airport_search(41.1499988, -95.91779)
```

```
40.6090011597 72.793296814


       ␣
↪---------------------------------------------------------------------------


       TypeError                                 Traceback (most recent call␣
↪last)

       <ipython-input-16-c71e6e37f601> in <module>
        15 #    pass
        16
   ---> 17 airport_search(41.1499988, -95.91779)


       <ipython-input-16-c71e6e37f601> in airport_search(latitude, longitude)
        11     print(latitude, longitude)
        12
   ---> 13     return pygeohash.geohash_approximate_distance(latitude,␣
↪longitude)
        14
        15 #    pass


       /opt/conda/lib/python3.8/site-packages/pygeohash/distances.py in␣
↪geohash_approximate_distance(geohash_1, geohash_2, check_validity)
        48
        49     # normalize the geohashes to the length of the shortest
```

```
---> 50     len_1 = len(geohash_1)
     51     len_2 = len(geohash_2)
     52     if len_1 > len_2:


TypeError: object of type 'float' has no len()
```

```
[ ]:
```