Assignment 3

January 17, 2022

1 Assignment 3

Import libraries and define common helper functions

[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

import pyarrow.parquet as pq

import fastavro
import pygeohash
import snappy
import jsonschema
from jsonschema import ValidationError, validate, SchemaError
```

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

[3]:
```
# Setting up the directory structure
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)

src_data_dir = current_dir.parent.parent.parent.joinpath('data/processed/
 ↪openflights//routes.jsonl.gz')

print(current_dir)
print(schema_dir)
print(src_data_dir)
```

```
/home/jovyan/dsc650/dsc650/assignments/assignment03
```

```
/home/jovyan/dsc650/dsc650/assignments/assignment03/schemas
/home/jovyan/dsc650/data/processed/openflights/routes.jsonl.gz
```

[4]:
```python
def read_jsonl_data():
    with gzip.open(src_data_dir, 'rb') as f:
            records = [json.loads(line) for line in f.readlines()]
    return records
```

[8]:
```python
# This exercise is to flatten the json files.
myRec = read_jsonl_data()
df_myRec = pd.DataFrame(myRec)
# Create separate dataframes for each of the nested dicts
# df_airline = pd.json_normalize(df_myRec['airline'])
# df_src_airport = pd.json_normalize(df_myRec['src_airport'])
# df_dst_airport = pd.json_normalize(df_myRec['dst_airport'])
# Concatinating them together to be flattened out
# df_final = pd.
 ↪concat([df_airline,df_src_airport,df_dst_airport,df_myRec['codeshare'],df_myRec['equipment'
 ↪axis=1)
# df_final.describe
```

[ ]:
```python
myRec[0]['src_airport']['latitude']
```

[9]:
```python
df_myRec.head()
```

[9]:
```
                                             airline  \
0  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
1  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
2  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
3  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
4  {'airline_id': 410, 'name': 'Aerocondor', 'ali…

                                         src_airport  \
0  {'airport_id': 2965, 'name': 'Sochi Internatio…
1  {'airport_id': 2966, 'name': 'Astrakhan Airpor…
2  {'airport_id': 2966, 'name': 'Astrakhan Airpor…
3  {'airport_id': 2968, 'name': 'Chelyabinsk Bala…
4  {'airport_id': 2968, 'name': 'Chelyabinsk Bala…

                                         dst_airport  codeshare equipment
0  {'airport_id': 2990, 'name': 'Kazan Internatio…      False     [CR2]
1  {'airport_id': 2990, 'name': 'Kazan Internatio…      False     [CR2]
2  {'airport_id': 2962, 'name': 'Mineralnyye Vody…      False     [CR2]
3  {'airport_id': 2990, 'name': 'Kazan Internatio…      False     [CR2]
4  {'airport_id': 4078, 'name': 'Tolmachevo Airpo…      False     [CR2]
```

1.0.1 3.1.a JSON Schema

[5]:
```
def validate_jsonl_data(records):
    schema_path = schema_dir.joinpath('routes-schema.json')
    with open(schema_path) as f:
        schema = json.load(f)
    validation_csv_path = results_dir.joinpath('validate_json.csv')
    with open(validation_csv_path, 'w') as f:
        for i, record in enumerate(records):
            try:
                ## TODO: Validate record
                jsonschema.validate(instance=record, schema=schema)
            except ValidationError as e:
                ## Print message if invalid record
                pass

records = read_jsonl_data()
validate_jsonl_data(records)
```

1.0.2 3.1.b Avro

[6]:
```
from fastavro import writer, reader, schema

def create_avro_dataset():
    schema_path = schema_dir.joinpath('routes.avsc')
    data_path = results_dir.joinpath('routes.avro')

    with open(schema_path) as f:
        myschema = json.load(f)

    records = read_jsonl_data()
    with open(data_path, 'wb') as f_out:
        writer(f_out, schema.parse_schema(myschema), records)

create_avro_dataset()
```

1.0.3 3.1.c Parquet

[7]:
```python
def create_parquet_dataset():
        parquet_output_path = results_dir.joinpath('routes.parquet')

        records = read_jsonl_data()
        df = pd.DataFrame(records)
        # df.head()
        table = pa.Table.from_pandas(df)
        pq.write_table(table,parquet_output_path, compression=None)

create_parquet_dataset()
```

1.0.4 3.1.d Protocol Buffers

[12]:
```python
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 not airline.get('name'):
        return None
    if not airline.get('airline_id'):
        return None
    if not airline.get('active'):
        return None

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

    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
        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)
        route.codeshare = record.get('codeshare')
        # route.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()))

records = read_jsonl_data()
create_protobuf_dataset(records)
```

1.0.5 3.2.a Simple Geohash Index

[13]:
```
# Assignment 3.2.a
def create_hash_dirs(records):
    geoindex_dir = results_dir.joinpath('geoindex')
    geoindex_dir.mkdir(exist_ok=True, parents=True)
    hashes = []
    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:
                ## TODO: use pygeohash.encode() to assign geohashes to the␣
 ↪records and complete the hashes list
                hash_ind = pygeohash.encode(latitude=latitude,␣
 ↪longitude=longitude)
                hashes.append(hash_ind)
    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'))

records = read_jsonl_data()
create_hash_dirs(records)
```

1.0.6 3.2.b Simple Search Feature

[ ]:
```
def airport_search(latitude, longitude):
    ## TODO: Create simple search to return nearest airport
    pass

airport_search(41.1499988, -95.91779)
```

[ ]:
```
records = read_jsonl_data()
for record in records:
        airline = record['airline']
print(airline)
```

[ ]:
```
# Open the compressed zip file
with open(src_data_dir, 'rb') as fread:
        # Now open the file to write to
        json_out = results_dir.joinpath('routes.jsonl')
        with open(json_out, 'wb') as fwrite:
                fwrite.write(gzip.decompress(fread.read()))
```