Assignment 3

November 3, 2021

1 Assignment 3

1.1 Nick Reardon: 10/27/2021

Import libraries and define common helper functions

[20]:
```
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

[21]:
```
records = read_jsonl_data()

# Check the records
records[0:1]
```

[21]:
```
[{'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']}]
```

1.2 3.1

1.2.1 3.1.a JSON Schema

[22]:
```
def validate_jsonl_data(records):
    schema_path = schema_dir.joinpath('routes-schema.json')
    with open(schema_path) as f:
        schema = json.load(f)

    # Had to create a CSV file because it wasn't specified as an argument
    validation_csv_path = results_dir.joinpath('validation-results.csv')

    with open(validation_csv_path, 'w') as f:
        for i, record in enumerate(records):
            try:
                ## TODO: Validate record
                # Check to see if it is in fact a JSON schema
                jsonschema.validate(instance = records[i], schema = schema)
                pass
            except ValidationError as e:
                ## Print message if invalid record
                print('Invalid Record{0} at place'.format(i+1))
                pass


validate_jsonl_data(records)
```

1.2.2 3.1.b Avro

[23]:
```
# import necessary packages
from fastavro import writer, reader, parse_schema
from fastavro.schema import load_schema

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
```

```
    # Use fastavro to read the schema
    schema = load_schema(schema_path)

    with open(data_path, 'wb') as out:
        writer(out, schema, records)


create_avro_dataset(records)
```

1.2.3 3.1.c Parquet

[24]:
```
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: Use Apache Arrow to create Parquet table and save the␣
 ↪dataset

            table = read_json(f)

        # Use PyArrow to write the parquet file
        pq.write_table(table, parquet_output_path)

create_parquet_dataset()
```

1.2.4 3.1.d Protocol Buffers

[25]:
```
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

    ## Look out for null values
    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('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')
    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')

        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)
```

1.2.5 3.1e File Size Comparisons

[30]:
```
# Compare the output sizes of all of the different files

# Specify the path
json_schema_path = schema_dir.joinpath('routes-schema.json')
avro_schema_path = schema_dir.joinpath('routes.avsc')
parquet_path = results_dir.joinpath('routes.parquet')
proto_schema_path = schema_dir.joinpath('routes.proto')

# Get size
json_schema_size = os.path.getsize(json_schema_path)
avro_schema_size = os.path.getsize(avro_schema_path)
parquet_size = os.path.getsize(parquet_path)
proto_schema_size = os.path.getsize(proto_schema_path)

# Create a dataframe
df = pd.DataFrame({'JSON': [json_schema_size], 'Avro': [avro_schema_size],␣
↪'Parquet': [parquet_size], 'Protocol Buffer': [proto_schema_size]})
df
```

[30]:

	JSON	Avro	Parquet	Protocol Buffer
0	4	3191	1975469	1073

[27]:
```
## Write Dataframe to comparison.csv
compare = results_dir.joinpath('comparison.csv')
with open (compare, 'w') as f:
    df.to_csv(f, header = True, index=False)
```

1.3 3.2

1.3.1 3.2.a Simple Geohash Index

[28]:
```
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 is True:
            latitude = src_airport.get('latitude')
            longitude = src_airport.get('longitude')
            if latitude and longitude:
                geohash = pygeohash.encode(latitude,longitude)
                record['geohash'] = geohash
```

```
                hashes.append(geohash)

    hashes.sort()
    trunc_letters = sorted(list(set([trunc[:3] for trunc in hashes])))
    hash_index = {trunc: [] for trunc in trunc_letters}

    for record in records:
        geohash = record.get('geohash')
        if geohash is True:
            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)
```

1.3.2 3.2.b Simple Search Feature

[32]:
```
def airport_search(latitude, longitude):
    ## TODO: Create simple search to return nearest airport

    #try:
    h = pygeohash.encode(latitude, longitude)
    dist = 0
    name = ''

    # Check every airport
    for u, record in enumerate(records):
        src_airport = record.get('src_airport', {})
        if src_airport:
            latitude2 = src_airport.get('latitude')
            longitude2 = src_airport.get('longitude')
            airport_name = src_airport.get('name')

            #if values not null
            if latitude2 and longitude2:
                h2 = pygeohash.encode(latitude2, longitude2)

                # Approximate the distance between the airports
                dist_n = pygeohash.geohash_approximate_distance(h, h2)
                if u == 0:
                    dist = dist_n
```

```
                else:
                    if dist > dist_n:
                        dist = dist_n
                        name = airport_name
    print(name)
    #except Exception as e:
    #    print(e)

    pass

airport_search(41.1499988, -95.91779)
```

```
Eppley Airfield
```

[ ]: