DSC 650 - Big Data

Ramsey King

January 16, 2022

Assignment 3

Import libraries and define common helper functions

In [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()]
    '''
    src_data_path = '/Users/ramse/Documents/GitHub/dsc650/data/processed/openflights/rou
tes.jsonl.gz'
    with gzip.open(src_data_path, '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 (https://storage.budsc.midwest-datascience.com/data/processed/openflights/routes.jsonl.gz)

In [2]:
```
records = read_jsonl_data()
```

3.1

3.1.a JSON Schema

In [3]:
```
def validate_jsonl_data(records):
    schema_path = schema_dir.joinpath('routes-schema.json')
    with open(schema_path) as f:
        schema = json.load(f)

    with open(schema_path, 'r') 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
                print(e, 'Given JSON data is INVALID.')

validate_jsonl_data(records)
```

3.1.b Avro

In [4]:
```
def create_avro_dataset(records):
    schema_path = schema_dir.joinpath('routes.avsc')
    data_path = results_dir.joinpath('routes.avro')
    with open('/Users/ramse/Documents/GitHub/dsc650/dsc650/assignments/assignment03/schemas/routes.avsc') as file:
        schema = file.read()
    schema = json.loads(schema)
    with open(data_path, 'wb') as out:
        fastavro.writer(out, schema, records)

create_avro_dataset(records)
```

3.1.c Parquet

In [5]:
```
def create_parquet_dataset():
    src_data_path = '/Users/ramse/Documents/GitHub/dsc650/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
    '''

    jsonl_path = '/Users/ramse/Documents/GitHub/dsc650/data/processed/openflights/routes.jsonl'
    with open(src_data_path, 'rb') as file:
        with gzip.open(file, 'rb') as writer:
            df = pd.DataFrame(writer)
            table = pa.Table.from_pandas(df)
            pq.write_table(table,parquet_output_path)
create_parquet_dataset()
```

3.1.d Protocol Buffers

In [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: Create an Airline obj using Protocol Buffers API
    if not airline.get('name'):
        return None
    if not airline.get('airline_id'):
        return None
    obj.airline_id = airline.get('airline_id')
    obj.name = airline.get('name')
    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
        src_airport = _airline_to_proto_obj(record.get('src_airport', {}))
        if src_airport:
            route.airline.CopyFrom(src_airport)
        dst_airport = _airline_to_proto_obj(record.get('dst_airport', {}))
        if dst_airport:
            route.airline.CopyFrom(dst_airport)
        codeshare = _airline_to_proto_obj(record['codeshare'])
        route.airline.codeshare = codeshare
        stops = _airline_to_proto_obj(record.get('stops', {}))
        if stops:
            route.airline.CopyFrom(stops)
```

```
        equipment = _airline_to_proto_obj(record['equipment'])
        route.airline.equipment = 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)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-6-3f43978928b0> in <module>
     92         f.write(snappy.compress(routes.SerializeToString()))
     93
---> 94 create_protobuf_dataset(records)

<ipython-input-6-3f43978928b0> in create_protobuf_dataset(records)
     72         if dst_airport:
     73             route.airline.CopyFrom(dst_airport)
---> 74         codeshare = _airline_to_proto_obj(record['codeshare'])
     75         route.airline.codeshare = codeshare
     76         stops = _airline_to_proto_obj(record.get('stops', {}))

<ipython-input-6-3f43978928b0> in _airline_to_proto_obj(airline)
     41     obj = routes_pb2.Airline()
     42     ## TODO: Create an Airline obj using Protocol Buffers API
---> 43     if not airline.get('name'):
     44         return None
     45     if not airline.get('airline_id'):

AttributeError: 'bool' object has no attribute 'get'
```

3.2

3.2.a Simple Geohash Index

In [27]:
```
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 c
omplete the hashes list
                encoded_lat_long = pygeohash.encode(latitude,longitude)
                hashes.append(encoded_lat_long)
                record['geohash'] = encoded_lat_long
    hashes.sort()
    # print(len(set(hashes)))
    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'))
    return list(set(hashes))

hash_list = create_hash_dirs(records)
```

3.2.b Simple Search Feature

In [74]:
```
def airport_search(latitude, longitude, hashes_to_check, distance_in_km):
    ## TODO: Create simple search to return nearest airport
    distance_in_m = distance_in_km*1000
    inputted_geohash = pygeohash.encode(latitude, longitude)
    geo_hash_distance_list = []
    counter = 0
    for hash_ in hashes_to_check:
        geo_hash_distance_list.append(pygeohash.geohash_approximate_distance(inputted_ge
ohash,hash_))
        if pygeohash.geohash_approximate_distance(inputted_geohash,hash_) == min(geo_has
h_distance_list):
            closest_hash = hash_
            closest_distance = pygeohash.geohash_approximate_distance(inputted_geohash,h
ash_)
    if min(geo_hash_distance_list) <= distance_in_m:
        for record in records:
            if closest_hash == record.get('geohash'):
                print('The closest airport is', record.get('src_airport').get('name'),
'at a distance of', closest_distance/1000, 'kilometers.')
                counter += 1
            if counter >=1:
                break
    else:
        print("No airport found in the distance specified.")

airport_search(41.1499988, -95.91779, hash_list, 19.546)
airport_search(35.23800, -80.91865, hash_list, 200)
```

```
The closest airport is Eppley Airfield at a distance of 19.545 kilometers.
The closest airport is Charlotte Douglas International Airport at a distance of 3.803 k
ilometers.
```