(CSCI 5751, SPRING 2020)

TEAM: ZEROS_AND_ONES

AYUSHI RASTOGI (5575842)
MOURYA KARAN REDDY BADDAM (5564234)
ROOPANA VUPPALAPATI CHENCHU (5595508)
SHRAVYA GADE (5592616)

Table of Contents

1. Overview

We are utilizing Yelp and NOAA weather datasets to probe into descriptive and predictive analytic business questions discussed below. Our primary objective is to engineer this data effectively in a NoSQL database (Cassandra), to make querying and analyzing optimal.

2. Business Questions/Objectives

2.1. Easy

1. What is the total review count for all the businesses?
2. What are the total number of windy days?
3. What is the minimum compliment count received by a user?

2.2. Medium

1. Find the top 5 rated businesses in a category at a given distance from a location (latitude and longitude coordinates)?
2. What is the probability of a new user reviewing a business in category A, given that he reviewed business in category B?
3. Which user's feedback is supported most by the community? (Based on upvotes given to the tip recorded by the user)

2.3. Difficult

1. Which kind of business is a user most likely to visit on rainy, sunny and snowy days?
2. Which user is most similar to user X? (in terms of his taste).

Below are the queries we used to answer these questions in Cassandra:

```
select sum(review_count) from business;
SELECT count(*) from day_cat WHERE cat='W' ALLOW FILTERING;
select MIN(compliment_count) from user_tip_comp_count_rank;
select * from user_tip_comp_count_rank where rank = 1 ALLOW FILTERING;
select * from busi_cat_day;
```

3. Datasets & Tech Stack

3.1 Datasets

Datasets	Yelp	NOAA
Format	JSON	CSV
Volume	Size: 8.69GB Sub datasets: Users, Reviews, Businesses, Check-ins, Tips	Rows: ~250 M Columns: 68
Description	Yelp dataset has attributes which describe various different businesses, reviews, details about users who reviewed those businesses.	Daily weather summaries with precipitation, air temperature, snowfall, sunshine from all stations in a region.
Source	Yelp dataset website[1] https://www.yelp.com/dataset	NOAA weather daily summaries [2] [https://www.ncdc.noaa.gov]
Collected for Years	The review sub dataset of the Yelp dataset ranges from the years 2004 – 2018.	(2004 - 2018) for all states in the United States

Description of each data file can be referred in the appendixYelp Dataset:

3.2 Tech stack

Technologies	Cassandra	Python
Functionality	Column Family Database – Data Storage	Libraries: Pandas, Boto3,Numpy, scikit learn, geom

Apache Cassandra:

Cassandra is a free open source column family based NoSQL database. It has the following capabilities:

1. It accommodates high availability and easy horizontal scaling by using partitioning and replication across multiple clusters
2. It has the ability to store up to 5TB of uncompressed data per node
3. The declarative query language Cassandra Query Language(CQL) is very similar to SQL and hence intuitive to use.

But there were a few cons in the database as well:

1. The newer versions are released every few months and its is hard to keep a track of dependencies and documentation
2. If the tables are too wide (have too many columns) fetching data rowise is not optimal

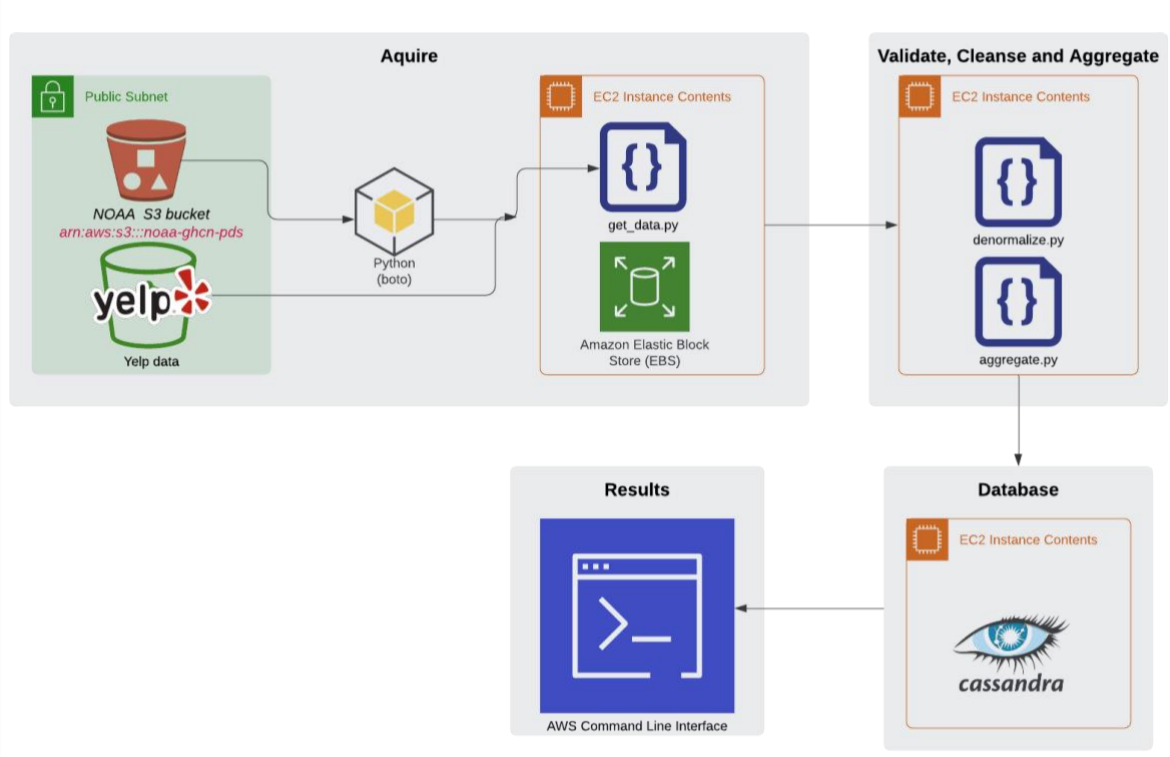
Python

Boto3: Official Amazon Web Services (AWS) SDK for python that provides an interface to AWS S3.

Python pandas: Extremely powerful python library used for data manipulation and analysis
Python scikit learn: Python library used for machine learning application

4. ETL Pipeline

The following chart is a high-level representation of the ETL pipeline for the Yelp and Weather dataset analysis.



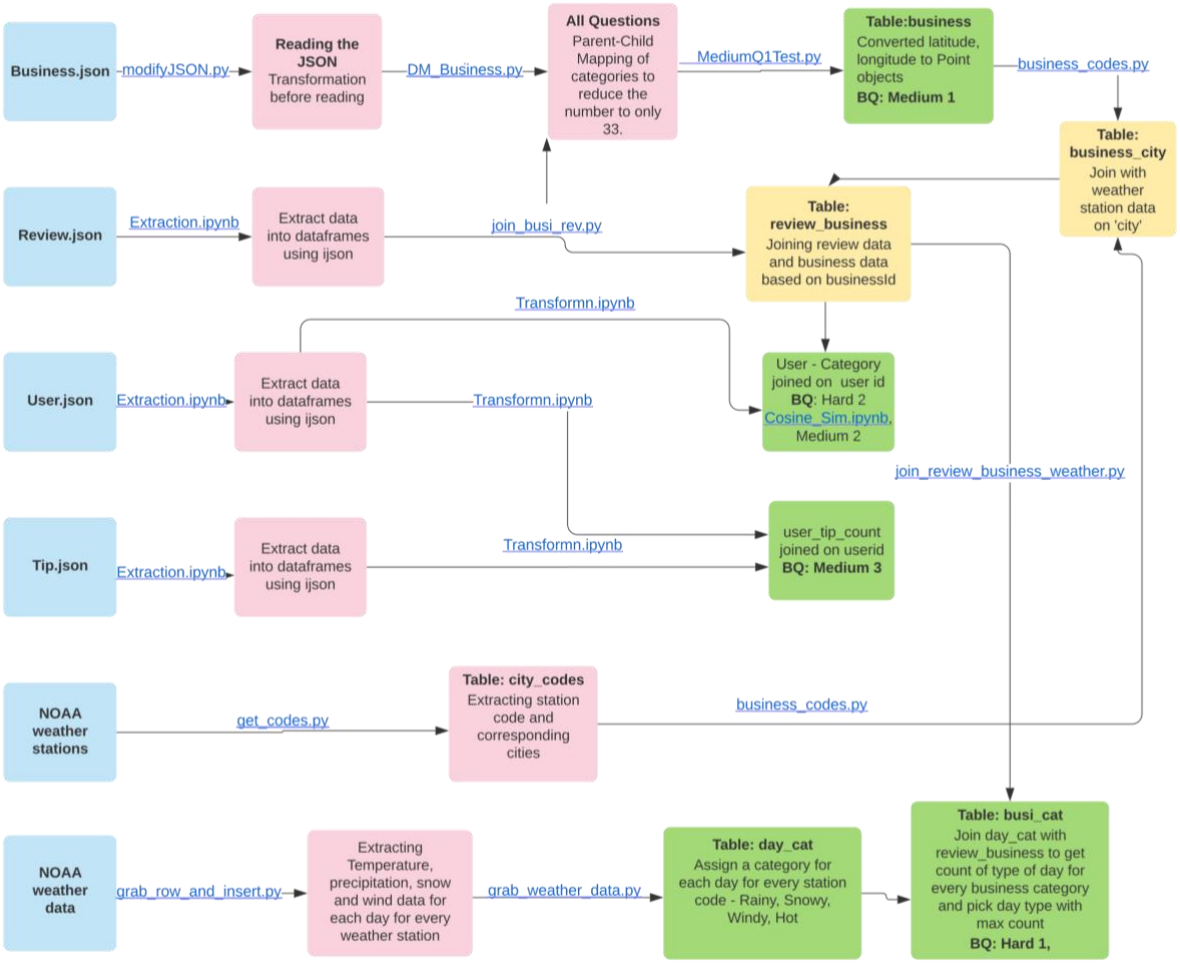
5. Data Processing

5.1. Data Preprocessing

1. Yelp data did not have the appropriate JSON array representation. We had to update the raw file by appending commas and then extract the data.
2. The business dataset had about 1300 categories which makes it impossible to analyze. Hence, to reduce the granularity, 33 categories were identified to which the existing categories can be mapped. Pandas library in Python was used for the transformation.
3. **Multiple levels of filtering for weather data** - The weather data was huge amounting to around 50GB and also had a lot of garbage values. We had to do multiple layers of filtering in order to extract attributes that were necessary for answering our hard question 2.

5.2 Data Transformation

Since Cassandra does not allow joins we used Python data frames to join the data as shown in the flow chart below. The code snippets are provided in the appendix.



Python Code for data transformation:

1. **Extracting Data**: modifyJSON.py, Extraction.ipynb, get_code.py, grab_row_and_insert.py
2. **Pre-Processing**: DM_Business.py, join_busi_rev.py, Transformn.ipynb, grab_weather_data.py, join_review_business_weather.py, business_codes.py,
3. **Answering Business Questions**: MediumQ1Test.py, Cosine_Sim.ipynb, review_cond_prob.py

The whole data can be found here in the repository. The description of each code snippet is given in the appendix.

5.3 Data Loading

We added the denormalized data to the database by connecting to Cassandra Cluster via python scripts as mentioned in appendix.

6. Data Models

We have used following data models to answer our business questions. We have explained in detail in our demo video.

6.1. Data Model 1

The denormalized dataset obtained by joining Business, Review and Weather datasets is used in Data Model 1. The *busi_cat_dat* column family is obtained by aggregating the count of reviews on weather type for each business category.

Column Family	busi_cat_day	day_cat
Domain	busi_cat : text, day_type : text Primary Key : busi_cat	station_code : text, date : date, day_type: text Primary Key : station_code , day
Index	N/A	N/A

6.2. Data model 2

Column Family	user_category
Domain	user_id : text, <category_name> : double Primary Key: user_id

6.3. Data model 3

Column Family	business
Domain	business_id : text, name : text, address : text, latitude : double, longitude : double, stars : double, review_count : varint, *cat_<name> : text Primary Key : business_id
Index	cat_<name> (All categories)

6.4. Data model 4

Column Family	user_tip_comp_count_rank
Domain	user_id : text, compliment_count:int, rank : int Primary Key : user_id

8. Challenges Faced

1. **Reading Big Data Chunks** - When processing huge JSON files (~1.5 GB) for Yelp data, reading the data entirely as a single chunk was challenging due to memory. In order to overcome this, we used the "ijson" library in python for reading data. It retrieves the data line by line iteratively and does not store the whole file as one entity in memory.
2. **Computational Inefficiency** - The user vector is of the order, 1637000*33. When computing user similarities, we need to store the whole data in memory which impacts efficiency of the process. We used sparse matrix representation to optimize memory given our data is sparse (70% sparsity). To precompute similarities for all the users the operation needs to run 1637000 times, giving O(n^2) complexity. This came out to be 18 days of run time (1 sec per user) after implementing multi threading and sparse representation. Hence we decided to compute similarity on need basis using python code by fetching user data from the database.
3. **Handicapped by the limited number of operations CQL provides** - We weren't able to use joins in CQL and our business questions required multiple joins, so we had to do a lot of denormalization at the application level (python)
4. **Version Conflicts on EC2** - We had Cassandra 2.x installed initially, but we realized that it didn't support any aggregate queries. So, we installed Cassandra 3.x. Now, this version of Cassandra supported python2 and few of us had python3 on our local, so we had to resolve version conflicts of few libraries which were not supported by python2. For example, we were using 'pygmaps package' initially for visualizing the distances among businesses. But, this wasn't supported on python version 2, so we later had to change it to 'gmplot package'.

5. **Collaborating distantly with teammates** - It was challenging to coordinate with teammates distantly. In the end, however, we learned to coordinate and come to a consensus through zoom meetings and shared docs.

9. POC Learnings

1. **Integration of Geomesa with Cassandra** - We had initially thought that we would use Geomesa along with Cassandra for answering our Medium Question 1. We were able to install it also, however, due to lack of more documentation or open-source support for using it we weren't actually able to implement it in our final POC. We had to work around and use Harvesine Distance for calculating the distance and then use the python 'gmplot package'.
2. **Designing column family for 1000+ distinct categories** - We had designed our data model such that each category should be a separate column. But, while analyzing the data, we observed that there were a total of 1300+ business categories. We realized that Cassandra's efficiency goes down when the number of columns are too large and hence bucketed our categories.
3. **Need for Indexing** - When we started to filter our data we realized that this operation was slow. We, therefore, created indexes for all 33 categories in the business table to speed up the READ operation.
4. **Unable to use aggregate queries due to missing replication in DB** – The initial idea was to implement DB replication, but due to lack of time we couldn't. We realized that DB replication is essential to perform aggregation queries efficiently on the huge amount of data in CQL.
5. **New Tech Stack for all Teammates** - We were all new to NoSQL tech stack. During the course of the project, we were able to learn Cassandra and Amazon EC2. Few of us were not comfortable with UNIX commands, but during this project, we got a hang of it.

Appendix

1. Yelp Dataset:

Business

Description: The dataset provided by Yelp for yelp data challenge which contains the details of the businesses included in the dataset. The data contains about **209,393** business names, precise location, review stars, categories and attributes related to the business. There is a business ID field that is used in all other sub-datasets like Review and Checkin to link back to the business table. The category field has many diverse categories and the total number of categories among all businesses came to a total of 1300 which resulted in too many overlaps between businesses.
Format: JSON
Scale: 138 MB file with 209,393 entries
Attributes:

Field	Type	Description	Sample Value
business_id	text	Unique ID for business	tnhfDv5Il8EaGSXZGiuQGg
name	text	Name of the business	**Garaje**
city	text	Manually entered city by the business owners	San Francisco
categories	List	List of categories the business belongs to	["Mexican", "Burgers", "Gastropubs"],
+11 more fields like latitude, longitude, attributes			

Review

Description: This sub-dataset contains the reviews given by users for various businesses. There are about **8,021,122** reviews in the dataset covering the details of date posted, by which user and for which business ID. Apart from the text included in the review we also have access to some tags assigned to the review.
Format:JSON
Scale: 5.36 GB

Attributes:

Field	Type	Description	Sample Value
review_id	text	Unique ID for review	tnhfDv5Il8EaGSXZGiuQGg
user_id	text	Unique ID for the user who gave the review	**Ha3iJu77CxlrFm-vQRs_8g**
business_id	text	Unique ID of the business for which the review was given	**tnhfDv5Il8EaGSXZGiuQGg**
stars	float	List of categories the business belongs to	4
date	text	Date when review was posted	**2016-03-09**
+2 fields with text in the review and number of votes for tags			

User

Description: The user.json sub dataset stores the details of the users in the platform along with the statistics of the user interaction with the platform. These values include the number of reviews given by the user and the year they joined yelp for every user. We were mainly concerned with the unique user IDs for answering our business questions and used this table to link the business table with review table to get the reviews by each user for a given business.

Format: JSON

Scale: 2.49 GB with 1637000 rows

Attributes:

Field	Type	Description	Sample Value
user_id	text	Unique ID for user	tnhfDv5Il8EaGSXZGiuQGg
name	text	Name if the user	**Mourya**

review_count	float	Total number of reviews given by user	**56**
useful	float	number of “useful” votes sent by the user	67
Average stars	float	average rating of all reviews	**4.32**
+15 fields compliment counts, yelp join date, friend user ids etc			

Checkin

Description: This dataset contains the details of checkins to a business. We observed that this dataset was very small and had fewer details (only business id and date and time of checkin). We understood that very few users used this feature of the app but if this dataset was rich it would have helped us answer many more business questions about correlation between the weather of a place at a given time and the users visiting a business.

Format:JSON

Scale: 420 MB

Attributes:

Field	Type	Description	Sample Value
business_id	text	Unique ID for business	tnhfDv5Il8EaGSXZGiuQGg
date	list	Dates and times of checkin	**2016-03-09**

2) Weather dataset

Station code with locations (ghcnd-stations.txt)

Description: This dataset contains 106200+ weather stations along with their details out of which the station code and the name were relevant for us.

Format: space separated CSV

Scale:10 MB with 106209 rows

Attributes:

Field	Type	Description	Sample Value
ID	text	Unique ID for each weather station	US1OKOK0071

NAME	text	Contains the city name as a part of the nomenclature	MINNEAPOLIS WB DWTN

Weather data for each year

Description: The yearly weather data is available at the link http://noaa-ghcn-pds.s3.amazonaws.com/csv/ with each file as <year>.csv. For eg for 2004 the file will be http://noaa-ghcn-pds.s3.amazonaws.com/csv/2004.csv. Each file has the data for all the stations in the world for everyday in the year and various fields.

Format: CSV

Scale: ~2GB for each year

Attributes:

Field	Type	Description	Sample Value
Station_ID	text	Unique ID for each weather station	US1OKOK0071
Date	text	Date of observation	"20040101"
Any attribute (TMAX,TMIN,AWND,SNOW)	text	Any of the attributes observed for a given code on a date	"TMAX"
Value of the attribute in previous column	float	The value of the field in previous column for a given station on a given day	"300"