UC RIVERSIDE
UNIVERSITY OF CALIFORNIA

Hopscotch 

Song Bai

Wei Chen

Rohit Chugh

David Feng

Anjali Ramchandani

Problem Overview

Many European cities have public pub crawls that serve as social gatherings for local tourists. They enable participants to meet new friends and become acquainted with new bars in a strange city. Often this requires prior research and knowledge of a new area, difficult for someone in a foreign place.

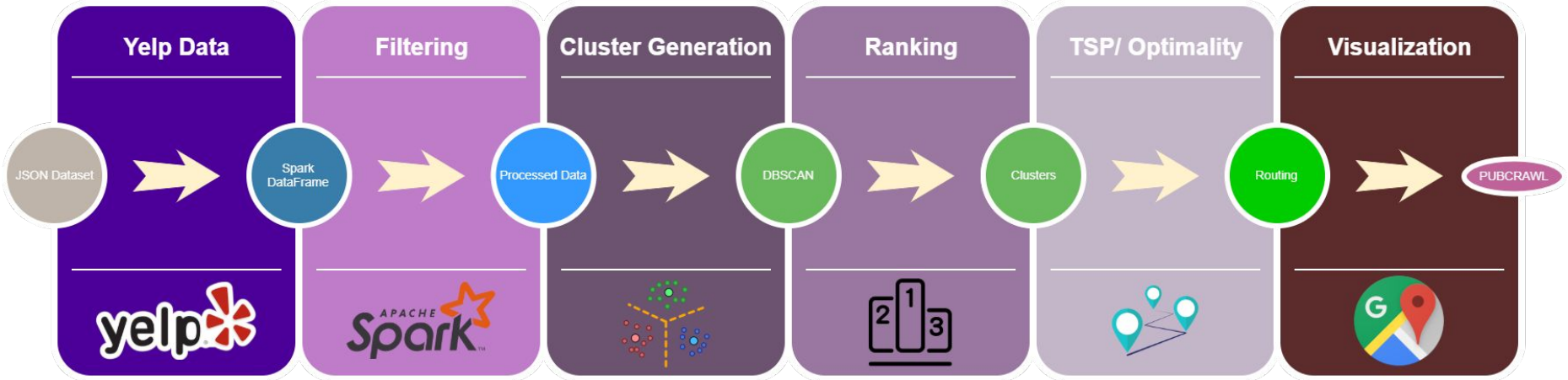
Goals

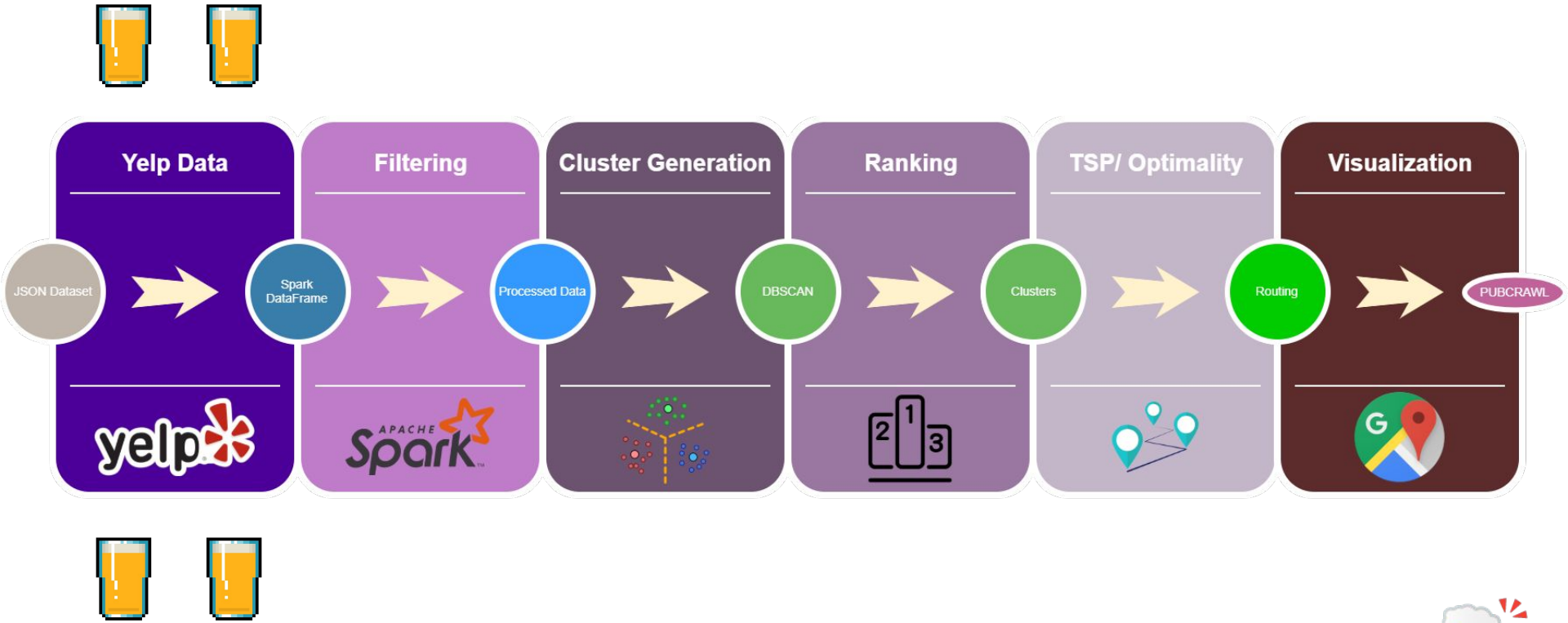
The goal of our project is to build a web app that **generates recommended routes** in new neighborhoods based on various inputs such as location, time duration, type of establishment, price range, busyness, popularity, and review score.

Our core dataset is the **Yelp Dataset**, which contains a trove of over 5M reviews, 100K businesses, and users, tips, check-in data. We will use this data to map out various **clusters** of establishments based on various **input signals** by the users.

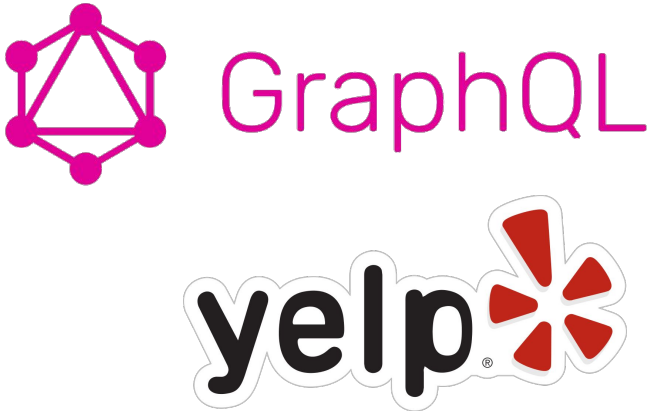
Further, using Google Maps API, we'll find out **optimal routes** between the establishments in these clusters. These routes will include travel options like walking, e-scooters, and rideshare services. Lastly, we render these routes on the map for end-user **visualisation**.

Project Pipeline




Yelp Data
Filtering
Cluster Generation
Ranking
TSP/ Optimality
Visualization
JSON Dataset
Spark DataFrame
Processed Data
DBSCAN
Clusters
Routing
PUBCRAWL
yelp
APACHE Spark
2 1 3
G

Dataset - Yelp



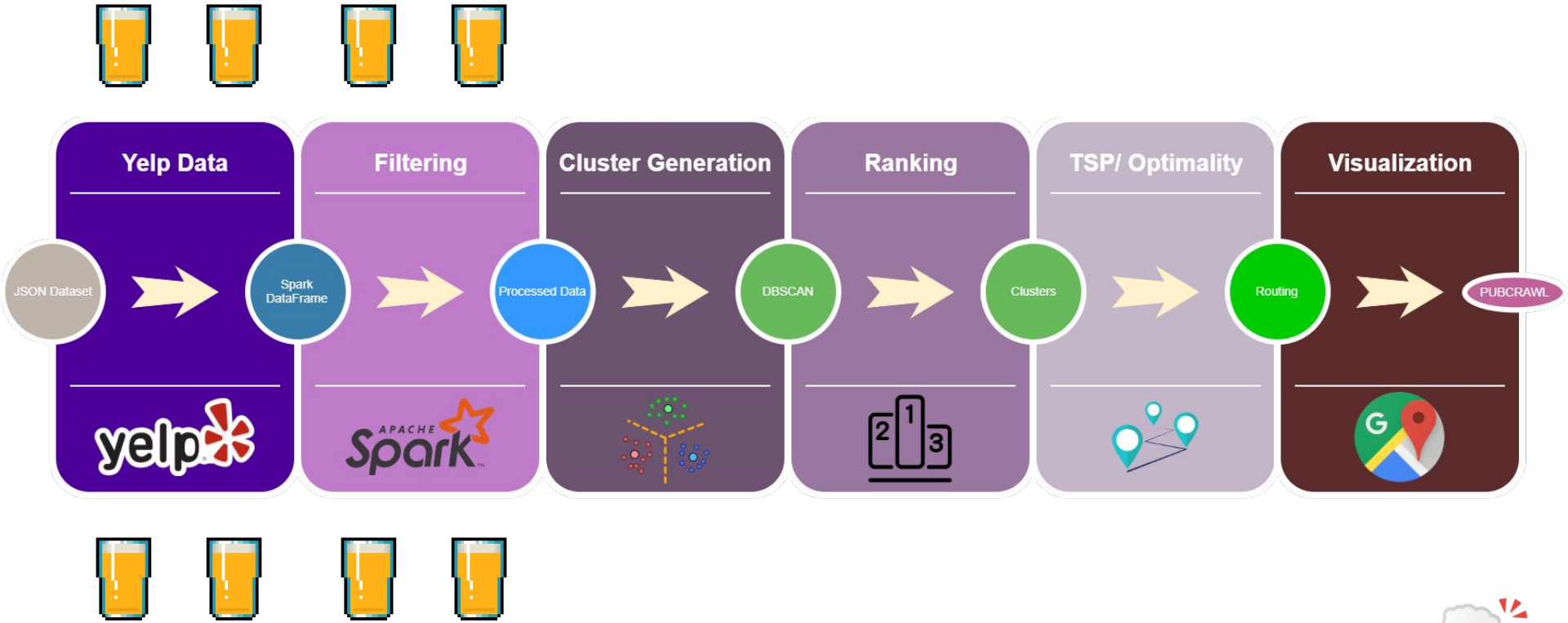
Problem:

- Limited number of available businesses
  - Only 19 in CA!

Solution:

Yelp API - GraphQL

- Scrape Businesses from Yelp using their latest API by relation: category(nightlife, restaurants, etc.)
- Discover businesses by Zip Code
  - convert from JSON to Spark DataFrame


Yelp Data
Filtering
Cluster Generation
Ranking
TSP/ Optimality
Visualization
JSON Dataset
Spark DataFrame
Processed Data
DBSCAN
Clusters
Routing
PUBCRAWL
yelp
APACHE Spark
2 1 3

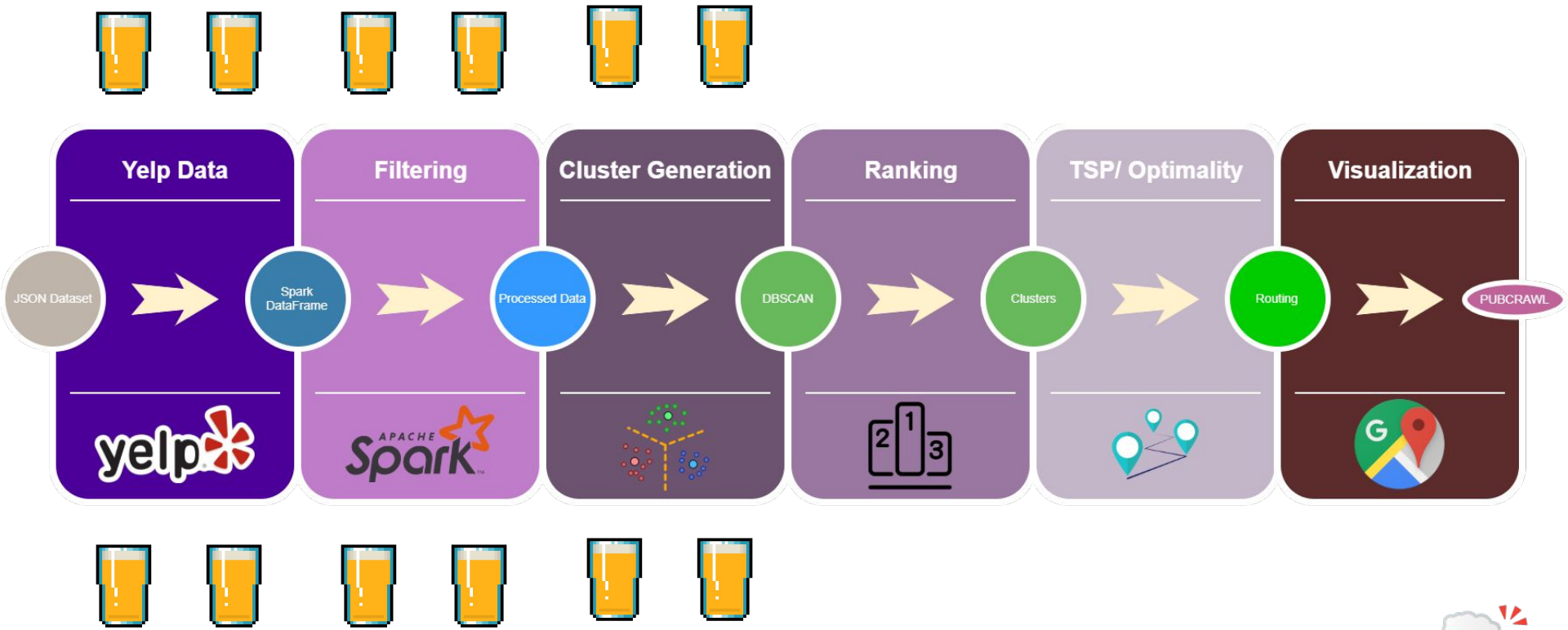
Filtering

The user provides the filters which server as the starting point of the algorithm.

Filters

- Location
- Open Hours
- Star Ratings


```
{
   "business_id": "tnhfDv5Il8EaGSXZGiuQGg",
   "name": "Garaje",
   "address": "475 3rd St",
   "city": "San Francisco",
   "state": "CA",
   "postal code": "94107",
   "latitude": 37.7817529521,
   "longitude": -122.39612197,
   "stars": 4.5,
   "review_count": 1198,
   "is_open": 1,
   "attributes": {
      "RestaurantsTakeOut": true,
      "BusinessParking": {
         "garage": false,
         "street": true,
         "valet": false
      },
   },
```



Yelp Data
Filtering
Cluster Generation
Ranking
TSP/ Optimality
Visualization
JSON Dataset
Spark DataFrame
Processed Data
DBSCAN
Clusters
Routing
PUBCRAWL
yelp
APACHE Spark
2 1 3

Clustering

- DBSCAN(Density-Based Spatial Clustering of Applicatio with Noise)

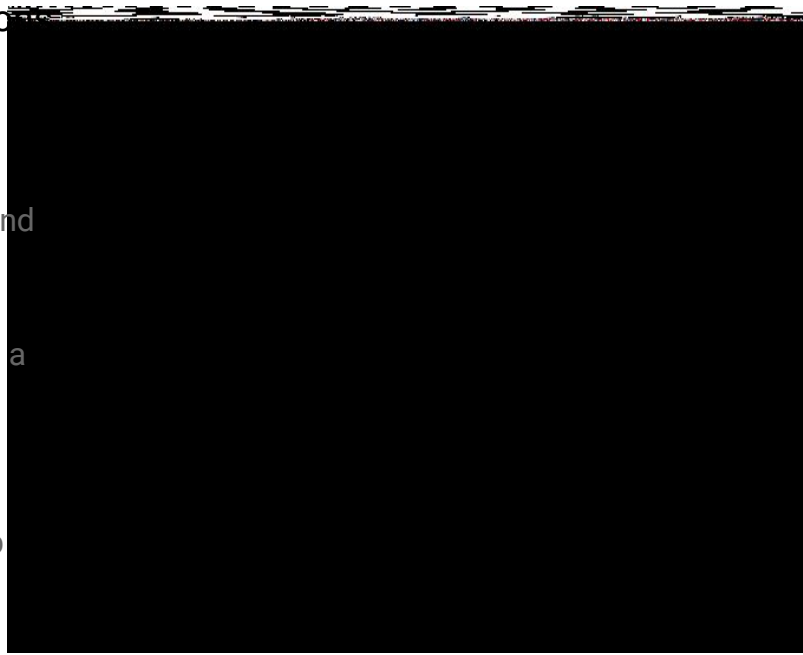
  Arbitrary select a point p

  Retrieve all points density-reachable from p reguading radius r(Eps) and MinPts.

  If p is a core point (has more neighborhoods than MinPts within Eps), a cluster is formed.

  If p is a border point (has fewer than MinPts within Eps, but is in the neighborhood of a core point), no points are density-reachable from p and DBSCAN visits the next point of the database.

  Continue the process until all of the points have been processed.



Clustering

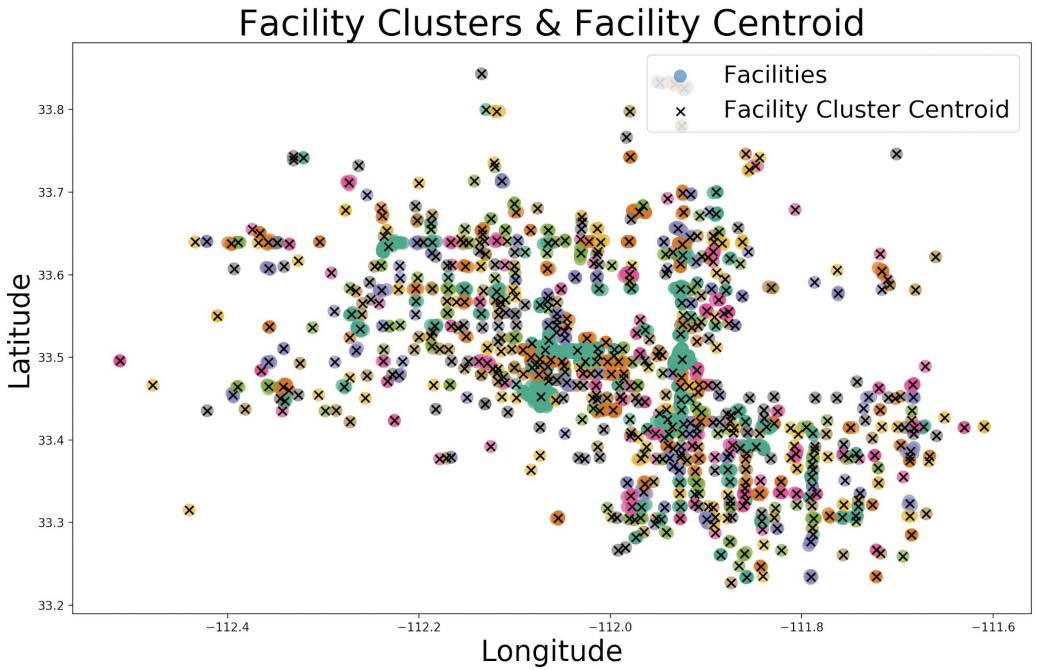
- Why DBSCAN?

It can find arbitrarily sized and arbitrarily shaped clusters quite well.

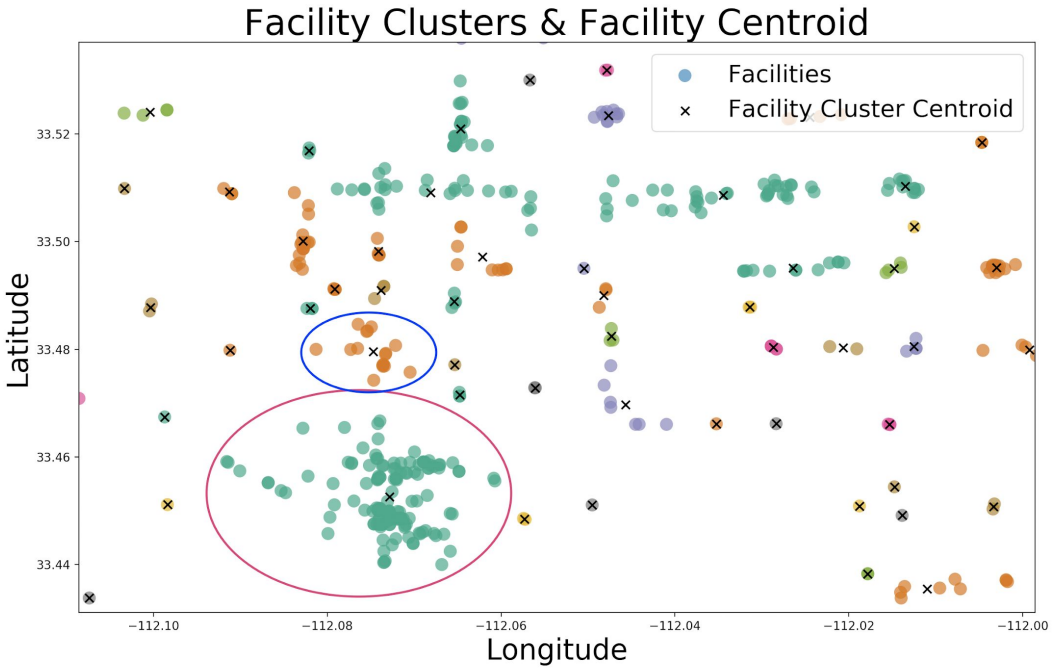
Does not require a pe-set number of clusters.

Identifies outliers as noises, unlike K-means and mean-shift which simply throws them into a cluster even if the data point is very different.

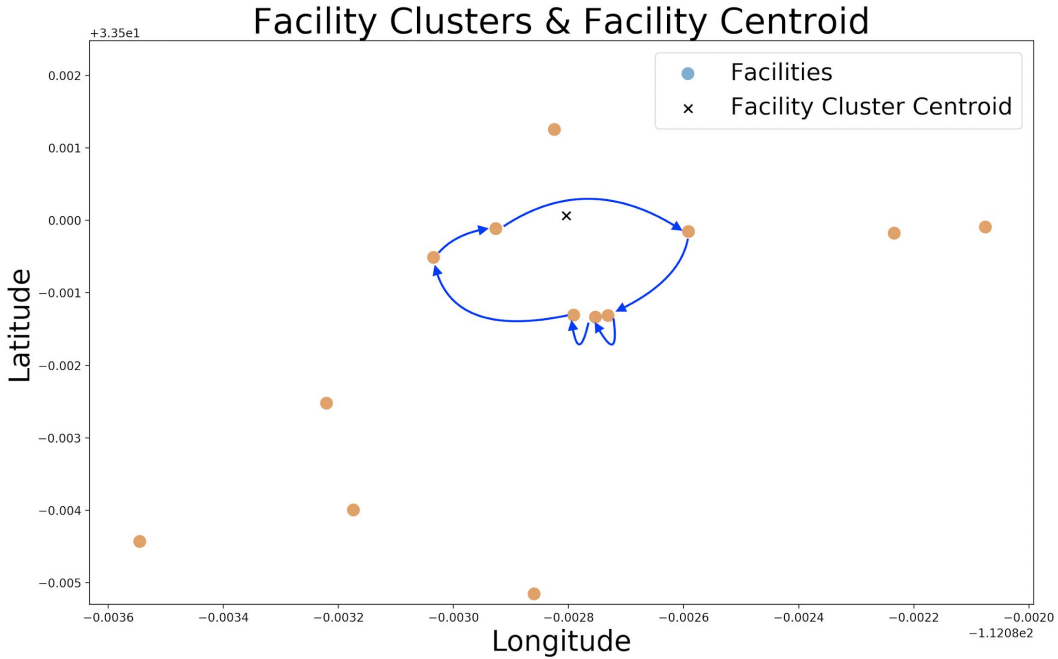
Clustering (using sklearn.cluster.DBSCAN)

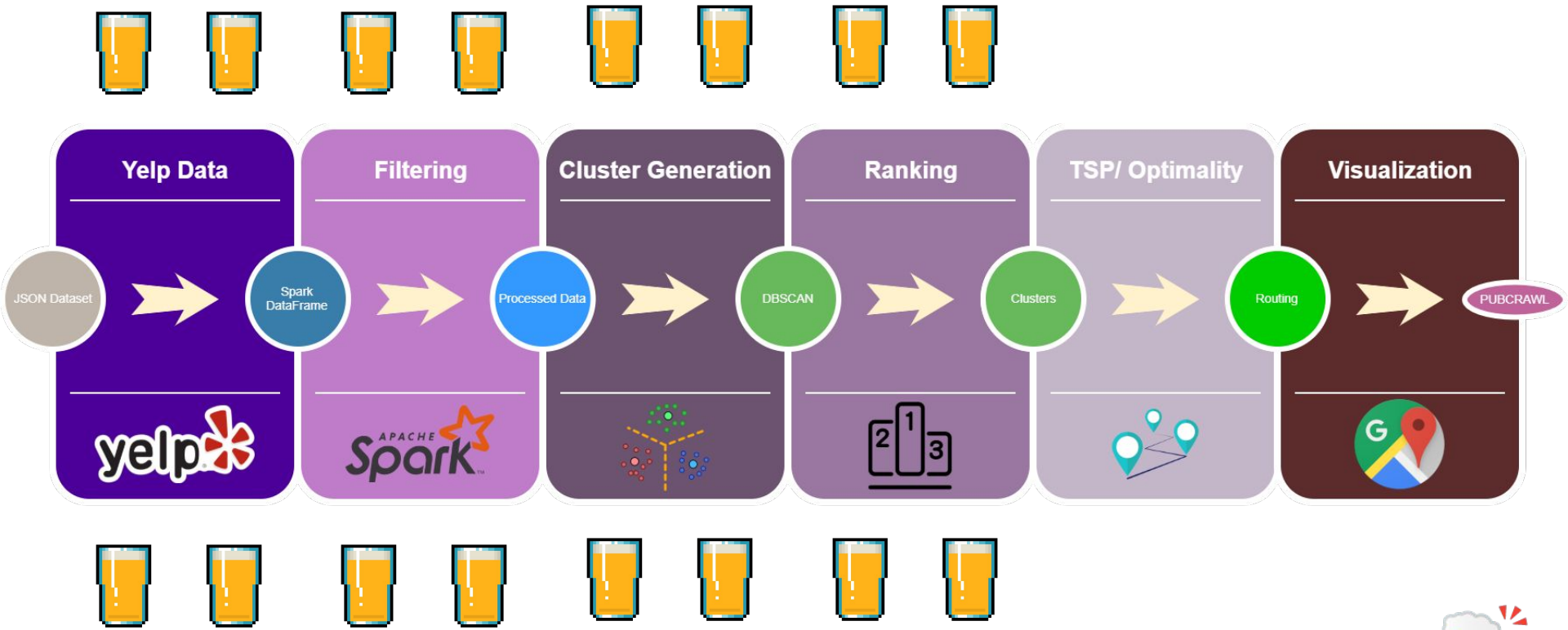


Clustering


Facility Clusters & Facility Centroid
Facilities
Facility Cluster Centroid
Latitude
Longitude
33.52
33.50
33.48
33.46
33.44
−112.10
−112.08
−112.06
−112.04
−112.02
−112.00

Clustering


Facility Clusters & Facility Centroid
Facilities
Facility Cluster Centroid
+3.35e1
0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.004
−0.005
Latitude
−0.0036
−0.0034
−0.0032
−0.0030
−0.0028
−0.0026
−0.0024
−0.0022
−0.0020
−1.1208e2
Longitude

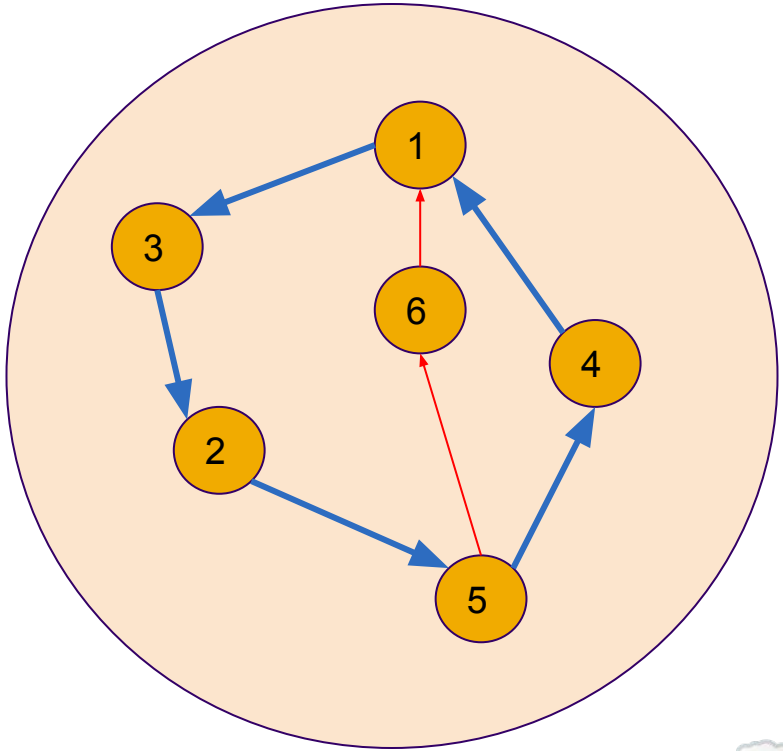

Yelp Data
Filtering
Cluster Generation
Ranking
TSP/ Optimality
Visualization
JSON Dataset
Spark DataFrame
Processed Data
DBSCAN
Clusters
Routing
PUBCRAWL
yelp
APACHE Spark
2 1 3

Ranking

Rank nodes in each cluster to get the best-n places to visit.

Factors to calculate rank:

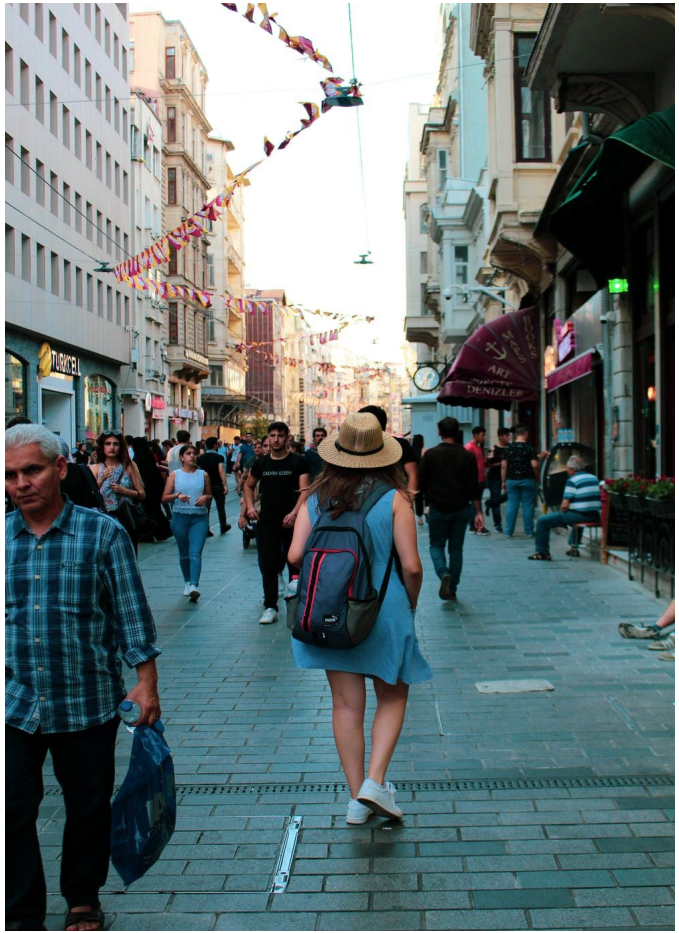
1. Ratings
2. Number of Reviews
3. Local Expert



Local Expert

Who?

"Local experts are the people who have expertise in a domain bounded by geographical area"



Local Expert

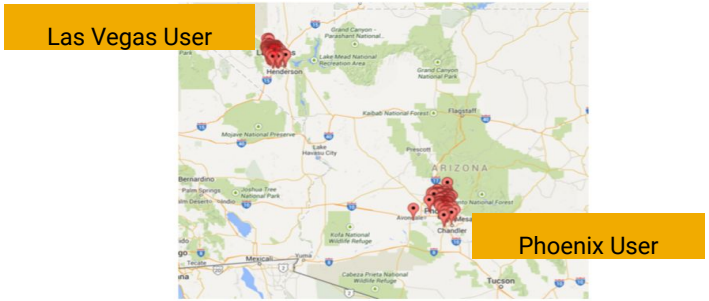
Why?

We want users to get the best *authentic* experience.

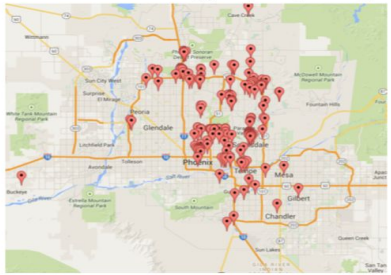
Local Expert

How do we find Local Experts?

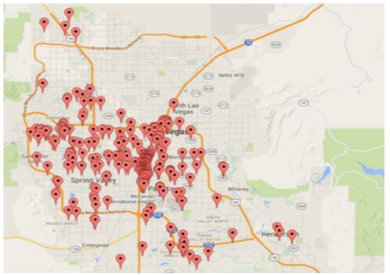
- Yelp does not give location of the users.
- The idea is that a user would visit the most businesses in the locality he/she lives.
- Using the businesses the user has visited/reviewed, we can approximate the location of a user.



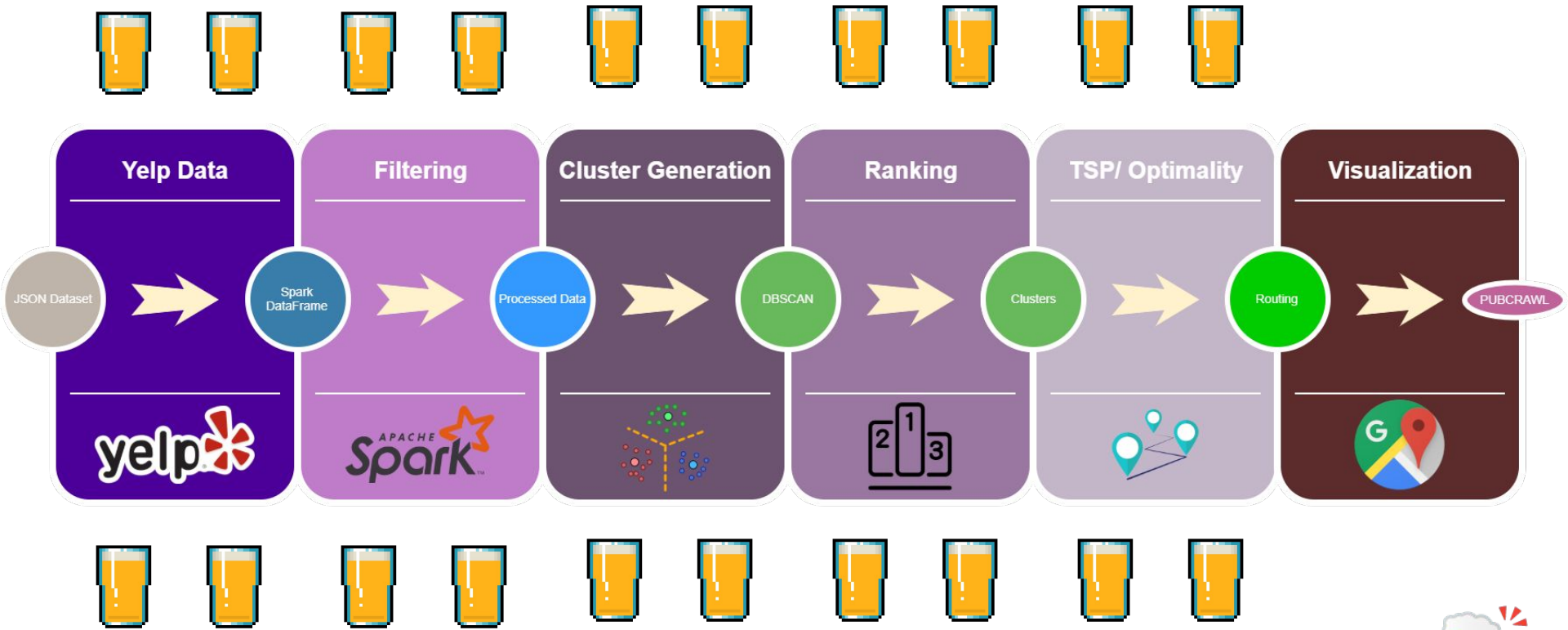
(a) All points, showing two clusters



(b) Zoomed in on Pheonix cluster

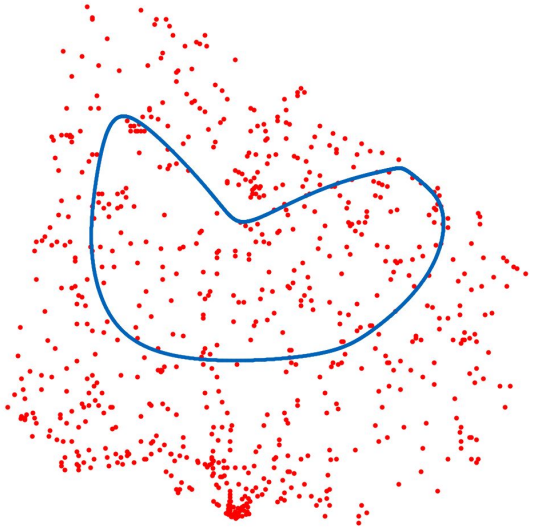


(c) Zoomed in on Las Vegas cluster


Yelp Data
Filtering
Cluster Generation
Ranking
TSP/ Optimality
Visualization
JSON Dataset
Spark DataFrame
Processed Data
DBSCAN
Clusters
Routing
PUBCRAWL
yelp
APACHE Spark
2 1 3
G

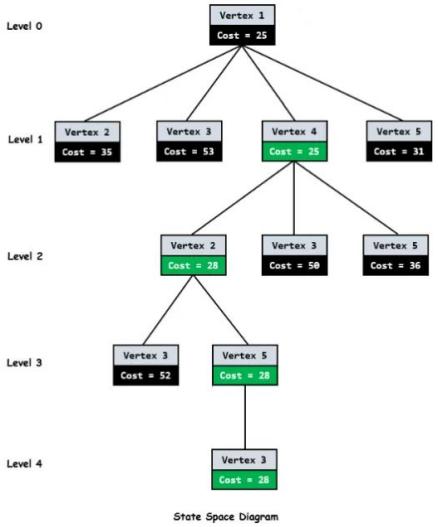
Traveling Salesman Problem

- Problem is NP-Hard
- How do we find best route given a limited window?
- Where is our start and end point? Do we need a cycle?
- How do we incorporate ranking into our path?

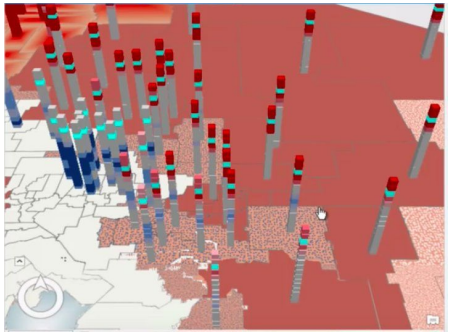


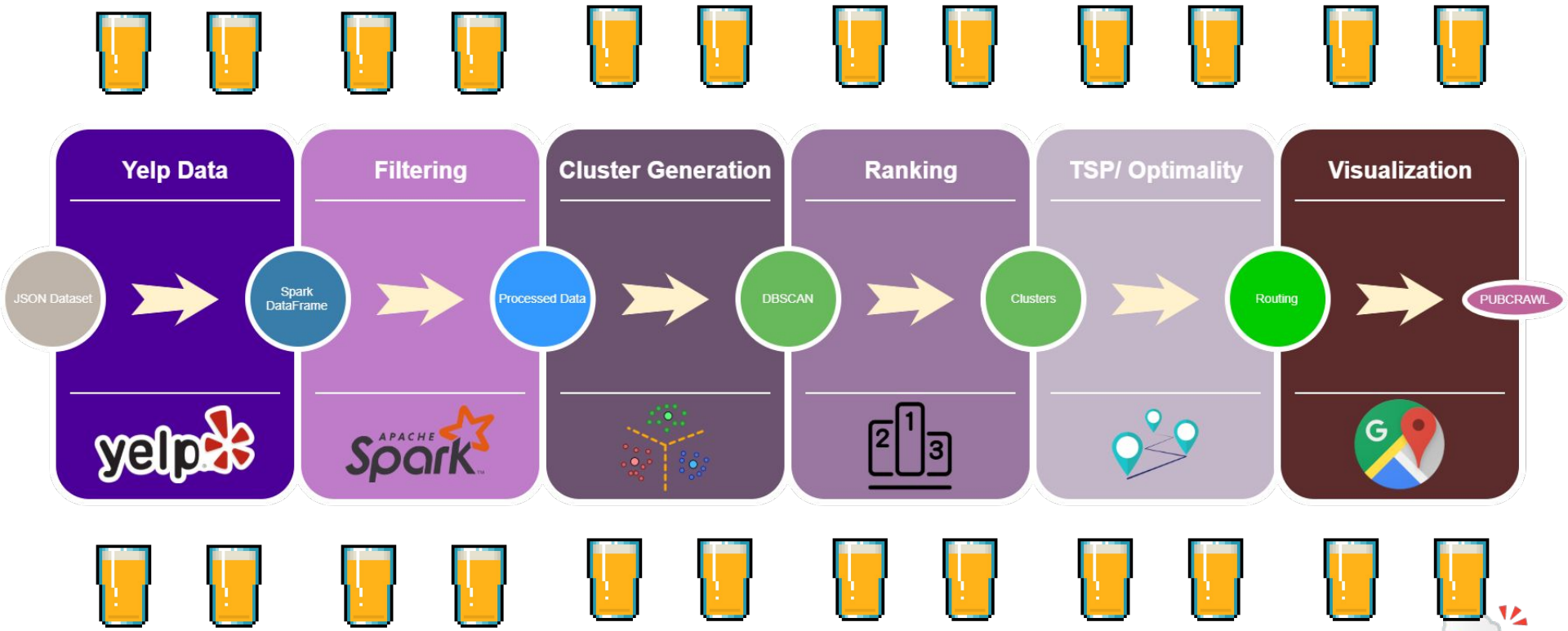
Heuristics and Optimality

- Branch and Bound
- Simulated Annealing
- Google Maps API - calculate travel distances:
  - By driving
  - By walking
  - By transit
- Google OR Tools - Combinatorial Optimization
- Ranking - incorporate a third dimension



State Space Diagram




Yelp Data
Filtering
Cluster Generation
Ranking
TSP/ Optimality
Visualization
JSON Dataset
Spark DataFrame
Processed Data
DBSCAN
Clusters
Routing
PUBCRAWL
yelp
APACHE Spark
2 1 3
G

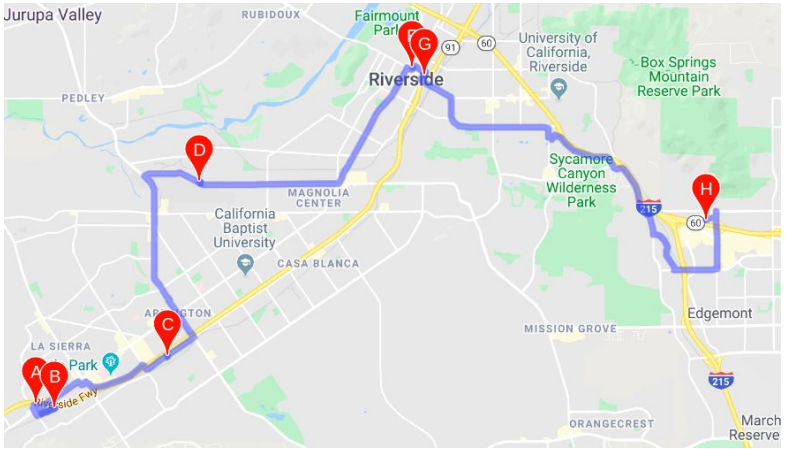
Visualization

- The route that travels through all the pubs will be shown
- The route contains starting point, destination point and several waypoints
- The format of points are in longitude and latitude


```
"rating": 4.5,
"location": {
  "address1": "800 N Point St",
  "address2": "",
  "address3": "",
  "city": "San Francisco",
  "zip_code": "94109",
  "country": "US",
  "state": "CA",
  "display_address": [
    "800 N Point St",
    "San Francisco, CA 94109"
  ],
  "cross_streets": ""
},
"coordinates": {
  "latitude": 37.80587,
  "longitude": -122.42058
},
```


Google Direction Service

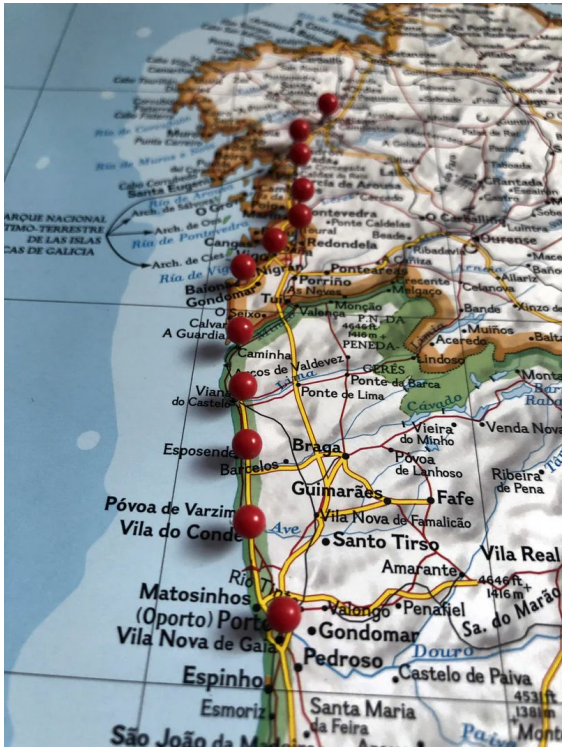
- Output can be visualized using Javascript in web browser
- It can respond on the fly whenever it gets some input
- The travel method can vary from walking to using public transportations



Future Goals

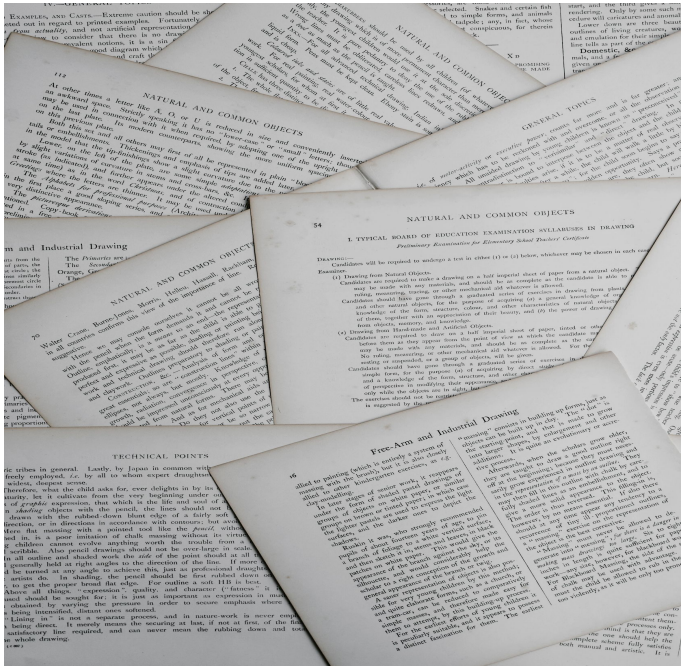
- User Interface
  - Represent establishments on Google Maps with route.

- Pre-calculate clusters and show them on graph in interactive mode.



References

1. Dataset - https://www.yelp.com/dataset
2. G. Clarke and J. Wright “Scheduling of vehicles from a central depot to a number of delivery points”, Operations Research, 12 #4, 568-581, 1964.
3. Kurz, Mary E. “Heuristics for the Traveling Salesman Problem.” *Wiley Encyclopedia of Operations Research and Management Science*, 2011, doi:10.1002/9780470400531.eorms0929.



CONCLUSION

Questions?