# Yelp Visualizations

Victor Hong, Tadj Mouhamed, Carlos Ng, Rohan Swaby, Norbu Tsering and Junjie Wu

***Abstract*— Yelp dataset includes multiple businesses and their user data such as user reviews, number of checkins, average business ratings and business categories for a large number of cities in the United States and around the world. Restaurants reviews and ratings are proven to be more central in the yelp dataset. In this regard, we use the dataset associated with restaurants to study the dynamics and trends in users eating habits in various cities in the United States. The discovery from this study is illuminating. To facilitate the understanding of the dynamics discovered, we represent our analysis of the underlying dataset in different visualizations.**

## I. INTRODUCTION

Yelp is a crowd-sourced aggregator of user reviews for various types of businesses. Along with user reviews, Yelp also collects important information about each business such as the address, nearby public transportation, website, and other tidbits like vegetarian options (namely for restaurants), parking, wi-fi, etc. One of the most popular uses for Yelp is for users to leave evaluations of their restaurant experiences, which other users will use to sway their decision making when choosing a spot to dine at. Although Yelp contains business information and reviews for many types of establishment, it is understandable to say that its prominent use is restaurant reviews. Through Yelp, users can see star ratings that previous patrons rated the establishment as well as a (possibly) lengthy text review. The text review can contain a myriad of information such as the reasoning behind the rating, recommended dishes, and other information about the restaurant. Along with user reviews, the Yelp website also contains attributes of the restaurant such as whether they offer parking, wi-fi, vegan options, if they take reservations or not, if they sell alcohol, etc. The abundance of information that Yelp offers makes it easy to understand why they beat other crowd-sourced review platforms such as Google.

The Yelp dataset [2] provided by Yelp concentrates a variety of the information they offer on the website and is open to anyone to use from their website. The Yelp dataset contains information regarding businesses from multiple states in the United States as well as a few other countries. For each business, they contain a business ID, location (longitude and latitude), star ratings with the attached date, attributes like the ones listed above amongst other pieces of data. Upon initial analysis of the dataset, there is potential for temporal visualization using the dated star ratings, as well as a geographical visualization using the longitude and latitude [6]. Restaurants can also be related to similar restaurants using attributes such as cuisine offered, vegetarian options, and other attributes [3]. As eluded, this visualization project will focus only on restaurants and not all businesses offered by Yelp. After scouring through the data, the conclusion was to focus on all restaurants in the state of Nevada. The choices made for the data chosen will be further elaborated in another section.

Beyond simply using the Yelp data as a tool for users to pick their next location to dine at, Yelp is also very useful for business owners. On a surface level, they can view all user ratings and reviews left for their establishment; this allows them to capitalize on positive points left by users as well as make changes based on repeated constructive criticism. Therefore, change in ratings over time is an important aspect of the data. However, good ratings without a lot of customers is moot; unfortunately, Yelp does not and cannot feasibly offer data for how many customers a restaurant gets. However, we can assume, since Yelp is such a widely used platform for leaving restaurant reviews, that the number of ratings/reviews left for a restaurant can be representative of the number of customers they get. However, using Yelp data, tools can be made to further enhance and influence a business owners decisions. While internal factors, such as the quality of the food or service, can make or break a restaurant (and can be represented by ratings/reviews), external factors can play a just as big or even bigger role in a restaurants success. For example, opening a Chinese restaurant in a neighborhood cluttered with high quality restaurant may not be a decision that is most conducive to success. Therefore, locational data must be used to in order to compare restaurants in an area by categories and attributes. By comparing the competition or pre-existing restaurants, business owners can make informed decisions towards where to open new restaurants or how to change their existing establishments.

The first visualization described, a temporal visualization, will be used to answer the question do recent ratings have an impact on the popularity of a restaurant. As stated above, since there is no actual measure of how many customers visit a restaurant, the number of ratings that month will be used as an indication of how popular the restaurant is. The temporal visualization will be implemented using a stacked bar graph with differing colors to represent the frequency of each level of ratings (how many 5 stars, how many 4 stars, etc.) in a time period selected by the users. Using this visualization, we hope to be able to analyze if there is a trend between star ratings and the number of customers. For example, a restaurant that receives great ratings in recent times should expect to see more customers while a restaurant that receives poor ratings should expect a decline in popularity.

The second visualization will be focused on locational visualization combined with the comparing restaurants using categories and attributes. Every restaurant will be plotted on a map of Nevada and the user will be able to select an

area by dragging a shape over the map. From the selected restaurants, a bubble chart is generated that represents all the present attributes [11]; the size of each bubble represents the frequency that each attribute shows up in all of the selected restaurants. The users can then select one or more attributes to see a heat map of how these attributes correlate with the ratings for each restaurant. Using this visualization, we can see how different attributes affect ratings for restaurants in areas that the users can select. As a tool for business owners, they can select different potential areas for new restaurants and analyze what types of restaurants do well. Alternatively, they can compare competitors in areas where their restaurant already exists to see how they differ and how these differences affect ratings.

## II. RELATED WORKS

Some work has already been done to provide business owners with insights using visualizations of the yelp dataset. Zheng et al [1] developed visualizations to find correlations between business ratings and secondary features in the yelp dataset [2]. However the visualizations they created are not temporal so business owners cannot see how changes in some variables affect businesses. Our work provides a temporal visualization so restaurant owners can see how changes in their restaurants ratings affect their business.

Zhiwei Zhang [3] documented his teams project using machine learning and visualization with the Yelp dataset. Their researched involved creating methods using machine learning to identify fake Yelp reviews and the keywords that are associated with fake reviews. To achieve this, they would need a way to display the most common words for each particular restaurant. The approach they chose to do this is through a word cloud. To generate their visualization, they used a support vector machine model for each restaurant. In our project, we will be focusing on each state in our word clouds to get a more general idea about the trending words in that state. In addition to the word cloud the team also had visualizations using maps that displayed the locations of each restaurant and the income level for that location. The use of a map is an effective choice for this visualization as both the restaurants and income level can be compared through location. For our project, we will have a slightly different approach where the location on a map will be used as an input for the word cloud associated with that location. Aside from the word cloud in this research, many of the other visualizations are bar graphs and other static mediums of visuals as the main focus of this research revolved around the machine learning aspect of detecting fake reviews.

In the paper Clustered Layout Word Cloud for User Generated Review by Ji Wang et al [9], more in-depth research on word clouds are conducted where the questions they attempted to answer include how semantic information of grammatical dependency graphs be embedded into word clouds and if the embedded semantic information influences user performance. The motivation behind this research is to allow for more efficient ways to present large amounts of user reviews for evaluations. To do this, they applied a natural language processing technique to the reviews to generate a semantic graph layout. Then using this graph layout, the LinLogLayout algorithm was used to create a force-directed graph layout. Finally a word cloud generation approach is applied to the force-directed graph layout [9]. Since they also used the academic Yelp dataset, this paper can help us decide our approach to creating the word cloud visualization. It also allows us to set effective parameters when creating the visualization as the paper contains research data of the different types of word clouds that were most efficiently processed. The conclusion of their research was that the clustered layout word cloud was more effective than the random layout word cloud for users to use and complete tasks with.


Fig. 1: word cloud

Another research that could help our project is by M. Tata [4] described a query driven system which helps business owners, investors and potential business owners make informed decisions based on analysis and visualization. The paper tries to answer the following questions: 1) What factors influence the success of a restaurant, based on customer reviews? and What areas are good for starting a business of would be best for investors? Based on these questions the paper defined its end users, users who will find their application useful. These users are, Current Business Owners and Future owners and/or Investors, people who would like to invest or start a new business in the future. M. Tata[4] Also define the pipeline for their project, which follows the pipeline for our project. The pipeline is as follows. First clean the relevant data present and store in a new database. Then Identify relevant data that could be useful to answering the research questions. Then finally performing data analysis on the data comparing factors that may affect the success or failure of a business with the average star rating of that business. These questions where answers by using visualizations such as bar charts showing correlations between different attributes, such as wifi availability and how that may affect rating. The M. Tata[4] stated that wifi played an important role in how ratings are affected. Simply because most restaurant businesses are located in the city and so are other businesses therefore people who leave their offices for lunch break will usually go the restaurants with wifi service.

Our project requires us to do some machine learning in order to develop a projection for our visualization. B.Hood[5] describe methods for inferring future business attention using regression models and sentiment analysis of reviews. To be more specific this paper tries to predict the number of reviews that a business should have at a particular time and also predict how many review for the next six months. To do this, The paper described ways to generate sets of features to use in their models. The methods are, simple manipulation of a given data and sentiment analysis on user-provided review. One of the ways this was done was creating a temporal prediction, after Principal Component Analysis (PCA) was done one the data a Temporal Prediction was made. The model was used for the following: predict the number of reviews that will be received for that business during a 6 month time period starting from a specific target date.

Ramesh et al used Yelp data in attempt to answer three research questions in similar vein to what we hope to accomplish [6]. The three questions are when, what, and where people prefer to eat. Each question was answered using an individual visualization for three visualizations in total. To answer the answer of when customers prefer to eat, they did a temporal visualization which mapped number of check-ins to a heat map across time and day of the week. This allowed them to see what time customers preferred to eat on which day of the week. From this visualization, they were able to determine that depending on the city, people had different preferences of when to eat; for example, in Vegas, check-ins were more prominent late night and on the weekends whereas in college towns check-ins were more common early evening on weekdays. To answer the question of what people like to eat, a bubble chart was used to show the popularity of each type of cuisine in each city. Larger bubbles meant a larger representation of that cuisine in the selected city. They also used Pearson correlation score to find cities with similar preferences in cuisine. To answer the question of where people like to eat, they used the Google map API and mapped restaurants on it with colored markers to represent star rating.

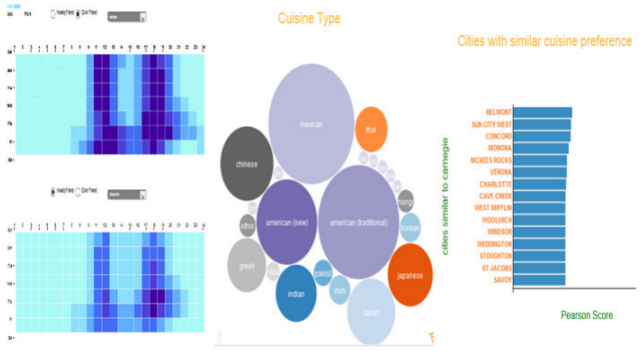


Fig. 2: word cloud

Overall, their visualizations can be used to make educated business decisions for potential restaurant owners when they want to decide on where to open a restaurant and what cuisine to sell.

Fan et at [7] paper talked about a way of predicting businesss star rating base on text alone. This was done by analyzing pure text review for each restaurant and compute their word count frequency. Using this info a machine learning model was used to compute the RMSE for each of those model. It turns out that a linear regression between the most frequent word and most frequent adjective are correlated. This can be use full for our word cloud.

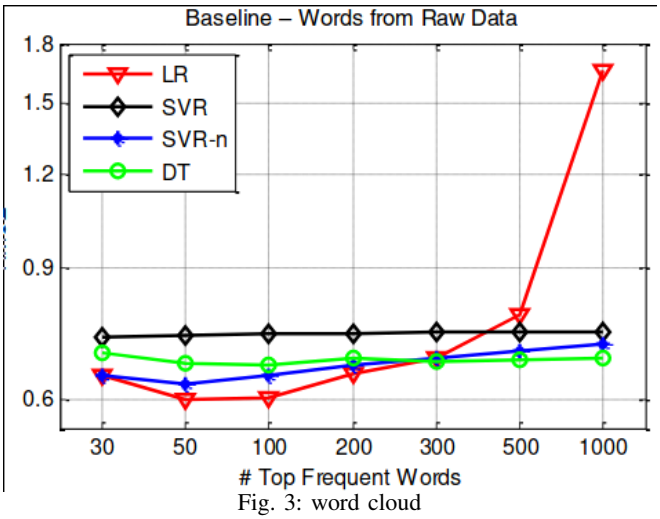


Fig. 3: word cloud

Akhila et at [8] paper talked about a way of visualizing yelp ratings aimed at business owners. This visualization tool will allow owner analyze the review for their store and gain insight across similar business. This is similar to our paper in a way that we want to help user get a deeper understanding of the reviews they see for a restaurants. The result of Fans research was an interactive display that compares your business within a given radius for a given category. Then it will generate multiple visualization.

In the paper titled Predicting Yelp Ratings Using User Friendship Network Information, Wenqing Yang , Yuan Yuan and Nan Zhang[10] did similar work with the yelp dataset. But this time some machine learning techniques were used to predict how users rate restaurants. The authors has a more broader dataset than we do. They had access to the friendship network of each user, that network is then represented in a undirected graph where each node represents a user and edges represent the friendship as shown in the table below.

| | |
|---|---|
| Number of nodes | 269231 |
| Number of edges | 986864 |

About 50% of the users have less than 120 friend, so to create the visualisation, the authors filtered out nodes with degree more than 120 and took a sample of 10% of the remaining nodes. In the plot of the friendship network, there

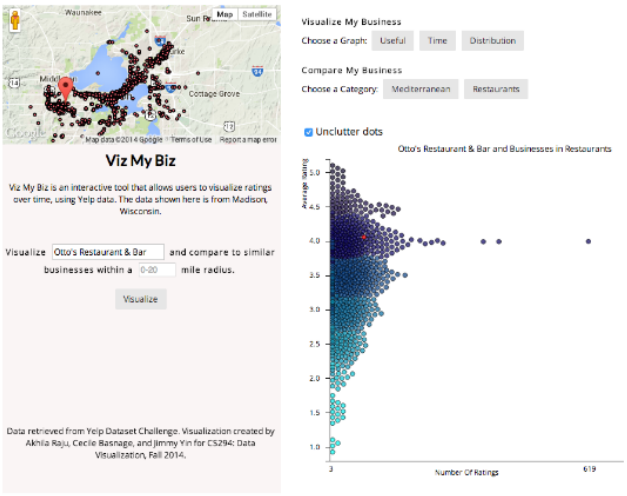



Fig. 4: word cloud

is a clustering pattern by most reviewed restaurants locations. Nevada for good reason clearly shows more activity than other states. To predict this behavior, the authors used the latent-model technique. An improved model is then used to do the prediction in a more efficient a accurate way . Even though this is not directly related to our projects, it illustrates the number of ways one can use the yelp dataset. The plot is shown below:

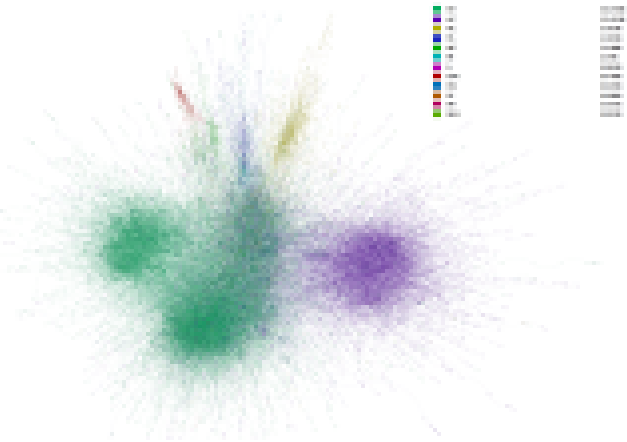


Fig. 5: word cloud

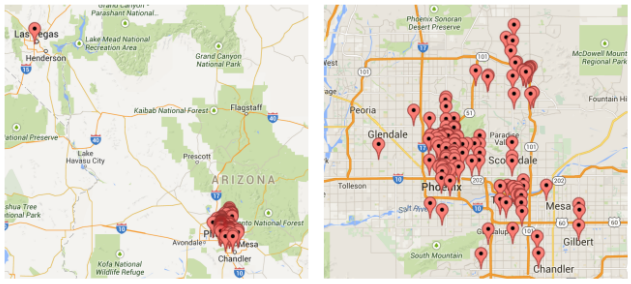
This paper titled Rendering Dorling Cartograms on Yelp Data by Weijia Jin and Jialiang Zhang [11] talked about dorling cartograms. Used in multiples representations, cartograms are used for mapping data. In this case, the authors used the yelp dataset to implement a Dorling cartogram. The cartogram focuses on the city of Las vegas. The city is represented using its zip codes. Each circle represents a zip code region in the city of Las Vegas, and both the radius and the color of the circle encode the average rating of all the businesses within the region.


Fig. 6: word cloud

The final result shows zip codes with the most rated restaurants and the poor rated one. From this, the user can be inform about what to expect depending on the area they live or spending their vacations.

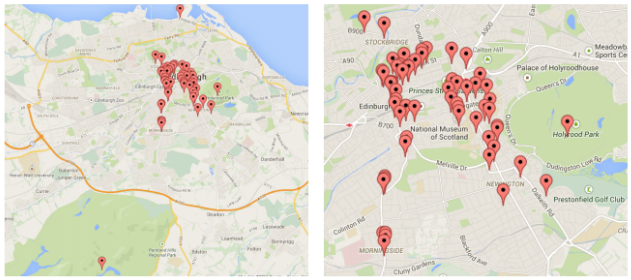
Tanvi Jindals project focused on connecting user reviews to their expertise in certain categories of establishments [12]. Unlike our project, his work did not specifically focus on restaurants but for all categories available on Yelp. He focused on user review data for specific users available via user ID provided by the Yelp dataset. The ultimate goal of his project is to find local experts in each category of establishment given location and category. By using the user ID, he can see all reviews left by the users; he categorized the reviews left by users by category of establishments the reviews are left for as well as other attributes such as average rating for each category of establishment. Therefore, users that have left a majority of their reviews for one certain type of category can be considered an expert of that category. Features of the user category data he processed also includes variance in ratings in that category and number of unique businesses reviewed. Given the list of experts, he now has to select experts local to the location given.

Yelp user data does not store address/location of the user so their location must be an inference made based on the reviews left by the user. First, business IDs are mapped to their locations by latitude and longitude. Reviews left by users are processed and user IDs are mapped with business IDs. Therefore, they can combine the data and produce a list of locations that each user has left reviews for. However, finding an equidistant point from all locations can be inaccurate due to reviews left during vacation or the user living in different locations during different periods of times. In order to find the locations of a user, Gaussian mixture model is used where a user can have multiple locations depending on large clusters of reviews in an area. Given a location and threshold distance, a local expert can be found. However, since this project doesnt directly deal with visualizations, it is not something we can draw inspirations from. Nonetheless,




(a) All points (b) Zoomed in

Figure 3.2: Location points for user1




(a) All points (b) Zoomed in

Figure 3.3: Location points for user2

Fig. 7: word cloud

the Gaussian mixture model can be something we employ if we want to process data in a similar manner.

## III. METHODOLOGY

### A. *Data Analysis*

The analysis in this project is done on the yelp dataset challenge. The data is collected in the first quarter of 2018. The dataset contains 174,567 business, 5,261,669 reviews, 3,911,218 check-in and attributes. The data includes different type of businesses, to simplify the task we decide to focus only on restaurants. Restaurants summed up to 134 209 in the United States and other countries involved. These restaurant are spread out in different categories among them are just-restaurants, cafes, Italian, Bakeries, middle eastern food, chinese, African, Japanese The table below shows the content of data regarding restaurants in all categories.

Table 1: Quantities of different categories of restaurants

| Restaurants(Categories) | Number of restaurant |
|---|---|
| Restaurants | 54 618 |
| Food | 24 777 |
| Bagels | 558 |
| Cofee | 5 936 |
| Pizza | 6 067 |
| Italian | 4 662 |
| Bakeries | 3 261 |
| Sandwitches | 6 345 |
| Asian Fusion | 1 658 |
| Cafes | 3 039 |
| Mexican | 4 105 |
| Seafood | 2 154 |
| Greek | 1 004 |
| Middle eastern | 1 029 |
| Mediterraneen | 1 593 |
| Comfort Food | 661 |
| Chinese | 3 987 |
| African | 142 |
| Specialty Food | 3 966 |
| Burgers | 4 558 |
| Japanese | 2 304 |
| Thai | 1 355 |

From here we created a separate table in the database that contains only these business categories. To do this we used the script below:

```
#ID procedure#
delimiter //
CREATE PROCEDURE get_id (OUT type
    CHAR(20))
BEGIN
    SELECT business_id from
        category where category=type;
#procedure#

#JOIN#
     select
         business.*,category.category
         FROM business inner join
         category ON business.id =
         category.business_id;
      - need to get the distinct
         business_id because each
         business have multiple
         category
      -filter the category table by
         id, then join it to business
       select DISTINCT business_id
           from category;
#JOIN#
```

To shrink and make much more focused, we cleaned up the data to include only United State based restaurants. We ended up with 31,840 restaurants across 13 metropolitan city across the U.S. To investigate the data state by state we decide to split the data into sub-tables, each table corresponding to a state. We use the script below to do this.

```
{DELIMITER //
CREATE PROCEDURE bar (IN state
    CHAR(20), bar_state CHAR(20))
BEGIN
    set @str = CONCAT("CREATE
        TABLE", bar_state, "like
        bar_NC;");
    PREPARE smt from @str;
    EXECUTE smt;
    DEALLOCATE smt;
    set @qr = CONCAT('INSERT INTO
        ', bar_state, ' (name,
        business_id, date, count)
        SELECT state.name,
        c.business_id, c.date,
        c.count from ', state, '
        left JOIN checkin c ON
        state.id = c.business_id;)';
    PREPARE smt from @qr;
    EXECUTE smt;
 END //
 DELIMITER ; }

{DELIMITER //
CREATE PROCEDURE filter_bar (IN
    bar_state VARCHAR(20), done_state
    VARCHAR(20))
BEGIN
    set @qr1 = CONCAT('CREATE TABLE ',
        done_state, ' like ',
        bar_state, ';');
    PREPARE smt from @qr1;
    EXECUTE smt;
    DEALLOCATE smt;
    set @qr2 = CONCAT('UPDATE ',
        bar_state, ' SET date =
        SUBSTRING_INDEX(date, ''-'',
        -1);');
   PREPARE smt from @qr2;
   EXECUTE smt;
   DEALLOCATE smt;

   set @qr3 = CONCAT('INSERT into',
       done_state,'select name,
       business_id, date, sum(count)
       as count from', bar_state,'
       group by name, business_id,
       date;');
    PREPARE smt from @qr3;
   EXECUTE smt;
END//}

UPDATE bar_state SET `date` =
    SUBSTRING_INDEX(`date`, '-', -1);
select name, business_id, date,
    sum(count) as count from bar_state
    group by name, business_id, date;
```

At first we decided to use all the data concerning the whole united states but the distribution of restaurants is not homogeneous. Restaurants are scarce in some states and concentrated in others. To make better use of the data we decided to focus on one state. After review of the dataset by state, we decide to use Nevada which contains a fair amount of data. The concentration of restaurants in nevada makes a good case to study. This data will be visualised in many ways to explore a specific question and each visualization will require different type of data on the state of Nevada. To visualize the temporal graph, we parsed the data and compiled each restaurant in the state of Nevada with its corresponding ratings. To do this we used the script below.

```
 DELIMITER //
CREATE PROCEDURE search_review (IN date
    VARCHAR(20), bar VARCHAR(20))
BEGIN
    set @qr1 = CONCAT('SELECT * FROM ?
        WHERE date = ?;');
    set @bar = bar;
    set @date = date;
    PREPARE smt from @qr1;
    EXECUTE smt USING @bar, @date;
    DEALLOCATE PREPARE smt;
END //
DELIMITER ;
```

This data is sent to the front end as a json file for further processing. To make each restaurants unique, we use the name and the address of restaurants as the key and the The Json file in question contains reviews for a specific dates. We used the python script below to construct the json file and make it easier to manipulate in the back-end.

```
import json
import sys
import os
# reading file
in_file = sys.argv[-1]
with open(in_file) as json_data:
   dictionary = json.load(json_data)

# declaring empty dict
output = {}

# calculating size of dict
len_dict = len(dictionary['rows'])
arr_stars = ["1 stars","2 stars","3
    stars","4 stars","5 stars"]

# parsing the data
for i in range(len_dict):
   name = dictionary['rows'][i]['name']
   address = dictionary['rows'][i]['address']
   stars = dictionary['rows'][i]['stars']
   year =
       dictionary['rows'][i]['date']['year']
   month=
       dictionary['rows'][i]['date']['month']
   key = name+'|'+address
   time = str(year)+'-'+str(month)

   if(key in output):
      #update new value
      current = output.get(key)
      which_stars = arr_stars[stars - 1]
      output[key][int(month)-1][which_stars]
          =
          current[int(month)-1][which_stars]
```

```
        + 1
    average =
        (current[int(month)-1][arr_stars[0]]
        +
        current[int(month)-1][arr_stars[1]]*2
        +
        current[int(month)-1][arr_stars[2]]*3
        +
        current[int(month)-1][arr_stars[3]]*4
        +
        current[int(month)-1][arr_stars[4]]*5)
        / 5.0
    output[key][int(month)-1]['average'] =
        average

else:
    #add new key
    value = [
          {"1 stars" : 0,"2 stars" : 0,"3
              stars" : 0,"4 stars" : 0,"5
              stars" : 0, "month":
              str(year)+"-01" ,"average" :
              0},
          {"1 stars" : 0,"2 stars" : 0,"3
              stars" : 0,"4 stars" : 0,"5
              stars" : 0, "month":
              str(year)+"-02" ,"average" :
              0},
          {"1 stars" : 0,"2 stars" : 0,"3
              stars" : 0,"4 stars" : 0,"5
              stars" : 0, "month":
              str(year)+"-03" ,"average"
              :0},
          {"1 stars" : 0,"2 stars" : 0,"3
              stars" : 0,"4 stars" : 0,"5
              stars" : 0, "month":
              str(year)+"-04" ,"average" :
              0},
          {"1 stars" : 0,"2 stars" : 0,"3
              stars" : 0,"4 stars" : 0,"5
              stars" : 0, "month":
              str(year)+"-05" ,"average" :
              0},
          {"1 stars" : 0,"2 stars" : 0,"3
              stars" : 0,"4 stars" : 0,"5
              stars" : 0, "month":
              str(year)+"-06" ,"average" :
              0},
          {"1 stars" : 0,"2 stars" : 0,"3
              stars" : 0,"4 stars" : 0,"5
              stars" : 0, "month":
              str(year)+"-07" ,"average" :
              0},
          {"1 stars" : 0,"2 stars" : 0,"3
              stars" : 0,"4 stars" : 0,"5
              stars" : 0, "month":
              str(year)+"-08" ,"average" :
              0},
          {"1 stars" : 0,"2 stars" : 0,"3
              stars" : 0,"4 stars" : 0,"5
              stars" : 0, "month":
              str(year)+"-09" ,"average" :
              0},
          {"1 stars" : 0,"2 stars" : 0,"3
              stars" : 0,"4 stars" : 0,"5
              stars" : 0, "month":
              str(year)+"-10" ,"average" :
              0},
          {"1 stars" : 0,"2 stars" : 0,"3
              stars" : 0,"4 stars" : 0,"5
              stars" : 0, "month":
              str(year)+"-11" ,"average" :
              0},
          {"1 stars" : 0,"2 stars" : 0,"3
              stars" : 0,"4 stars" : 0,"5
              stars" : 0, "month":
              str(year)+"-12" ,"average" :
              0}
       ]
    output[key] = value
    current= output.get(key)
    which_stars = arr_stars[stars - 1]
    output[key][int(month)-1][which_stars]
        =
        current[int(month)-1][which_stars]
        + 1
    average =
        (current[int(month)-1][arr_stars[0]]
        +
        current[int(month)-1][arr_stars[1]]*2
        +
        current[int(month)-1][arr_stars[2]]*3
        +
        current[int(month)-1][arr_stars[3]]*4
        +
        current[int(month)-1][arr_stars[4]]*5)
        / 5.0
    output[key][int(month)-1]['average'] =
        average


# writing output to file
out_file = "stars_done_"+str(year)+".json"
with open(out_file, 'w') as f:
   json.dump(output, f, indent = 4)
```

The next visualization consist of a map that locates all the restaurants in the Nevada. To achieve this visualization, the data was scanned and each restaurant is associated with its location coordinates (longitude, latitude). All the data is then passed to the front end to populate the map. To visualize the attributes related to different restaurants, a dorling cartogram is used where each bubble represent an attribute. To fulfil this representation we screened the yelp dataset and parse each restaurant with its attributes in json file sent to the front end. For many Visualizations we used the same json file and its to the back end team to filter and use the data interested for a specific visualization.

### *B. Front End*

*1) Temporal Visualization:* In order to generate the visualization for the user, D3 is used in conjunction with ReactJS. ReactJS is used to create the layout of the website to house the D3 visualization as well as update and pass data. The React component will mainly consist of dropdown menus that will allow users to customize their experience with the temporal visualization. The drop down menus will allow the users to select the starting month, starting year, and the number of months represented by each bar in the bar graph. The main React app stores the API link in a

variable and upon each new request to change either the month, year, or number of months, fills in the parameters to the API URL and fetches the new data. The newly fetched data is parsed into a JSON object and stored in a state of the React component. The array state that holds the data is then passed to the D3 React component as a prop. Each time the data is updated, the D3 component should update with regards to the new data as well. Originally, the temporal visualization will also employ a geographical visualization showing the map of Nevada which will also allow the user to select the restaurant, but the geographical visualization was re-purposed for another set of visualizations. Instead, restaurants will likely be chosen by drop-down menu instead.

We used a stacked bar graph in order to show temporal data using the Yelp dataset. Each bar of the stacked bar graph will represent the number ratings of a restaurant for a given number of months. The user can select how many months of data each bar represent. The bar is stacked and sectioned off by colors to show the quantity of each star rating. The overall height of the bar shows the total number of ratings for the time interval selected by the user. Unfortunately, given the limitations of the Yelp dataset, there are limited attributes that can be used to help us determine popularity of a restaurant. In the dataset, check-ins are only used to represent reservations, which may not be representative of all restaurant visitors. Under the assumption that the number of ratings/reviews is proportional to the total number of restaurant, we can use total number of ratings as a representation of how popular a restaurant is during that time period. Using the stacked bar graph, we can see if recent good ratings (4 or 5) can positively impact a restaurant's popularity (number of ratings in the following time interval) and if poor reviews (1 or 2) can negatively impact a restaurant's popularity. For example, if a restaurant receives a series of poor ratings, we can probably expect them to have fewer visitors and therefore fewer ratings. On the other hand, a series of positive reviews may make other customers to feel more inclined to visit and leave ratings. For an informed business owner, they can also use this as a tool to pinpoint a time period where they have made changes to their establishment and see how these changes affected the ratings as well as how it affected the popularity of their establishment. The tool will be versatile since it will allow the users to select a the start year, start month, and how many months of data each bar will represent. This will allow users to analyze short term effects of ratings as well as long term effects, although to a lesser degree.

D3 is used to realize the stacked bar graph using data received from the main React app. The D3 temporal visualization component receives data where each month has an integer value for each star rating as well as an average and total number of ratings. Each star rating is represented by a different color and bars of different colors are stacked on top of each other per month (or per month interval chosen by the user). The stacked bar graph for each time interval chose by the user will show the total number of ratings and the color distinction will show the distribution of each star rating. Hovering over each bar will also show the average rating for that month. This visualization shows two main things regarding the data: the distribution of ratings for the chosen time interval as well as the total number of ratings. This will help the user infer information such as how the distribution of ratings/average ratings over a time period influences restaurant popularity, as stated above.

*2) Geographical Visualization and Bubble Chart:* Another visualization we planned to use was a combination of a geographical visualization, a bubble chart, and a heat map. The geographical visualization will be used to map all restaurants in Nevada, which is the state that we chose to focus on. The map will receive data from the back end for every restaurant and each restaurant will be plotted on the map using longitude and latitude. The user will be able to circle off restaurants on the map to select multiple restaurants. The selected restaurants will return all the business IDs for the establishments, which will be used to generate the bubble chart.


Fig. 8: word cloud

The bubble chart takes in the business IDs and then returns a bubble chart based on the attributes related to all these restaurants. The size of the bubble is representative of how often the attribute appears in the restaurants that are selected from the map. For example, if a lot of restaurants in the selected area offer alcohol, then the alcohol bubble on the chart will be larger compared to other attributes. Conversely, if few restaurants offer wi-fi, then the wi-fi bubble will be small compared to the other attributes. The heat map will be implemented in conjunction with Pearson correlation or t-test performed by the back end. On the bubble chart, users will be able to select one or more attributes. If one attribute is selected then Pearson is used. If more than one are chosen, then t-test will be used. This is performed for each restaurant where rating is correlated with the attributes chosen and a

heat map is generated for the values derived from the Pearson correlation or t-test. For the attributes chosen, if a restaurant has good correlation with ratings, then it can be inferred that having these attributes have a positive contribution towards a restaurant's performance.

### C. Back End

The API was written in Java using Spring Boot which follows the MVC design pattern. We originally wanted to design the API using Django but we had a lot of problems with continuous deployment. To make sure everybody has access to the project, we wanted to setup a CI/CD pipeline on gitlab but Django was not supported whereas Spring boot worked very easily. Spring boot with its rapid application development features not only simplified the process but also provide many other benefit. In addition to spring boot, we setup the Maven tool. Maven gives the ability to better document, manage , describe dependencies in the project and set up a an automated test for each and every class. The project in overall is structured into multiple classes. These classes are grouped in two, main classes and the classes created to test the main classes. The main classes comprise the Api controller class where endpoints are created, the model class where the structure of the data to be return is declared and initialize and we also have the service classes and the map classes. Each visualization has a service class a map class and a testing class. The Api controller calls the correspondent function in the service class which then calls the corresponding function in the map classes. The actual manipulation of the data reside in the map classes. Each request from the front end is handled inter dependently among several classes and the resulting data is sent back to the front for visualization purposes. For the temporal visualization for example, the api controller calls a function named getTemporalModels which resides in the temporalMap class. Some elements of the TemporalModel class are used in the function getTemporalModels which return a set of data that is used to plot the bar chart. The same dynamic happens with other visualizations as well. A detailed analysis of the backend require us to go through each code, algorithm and try to give a precise documentation. At the begining, each request is handled by analysing the data and identifying certains attributes, or variables based on the request. First, the data concerning the request is downloaded and read using basic input ouput technics. Then we create a model for each for each data to manipulate. The model indicates the data structure we used for every data. For example, the model for the data used to compute correlations is shown below.

```
package com.seniorDesign.models;

import
    com.fasterxml.jackson.annotation.JsonProperty;
import lombok.AllArgsConstructor;
import lombok.EqualsAndHashCode;
import lombok.Getter;

@Getter
@EqualsAndHashCode
@AllArgsConstructor
public class CorrelationModel {
   @JsonProperty("business_id")
   String businessId;

   @JsonProperty("attribute")
   String attribute;

   @JsonProperty("correlation")
   double correlation;
}
```

Here we used the Lombok library which automatically creates getters and setters method. We rely on those methods to access the values we need from other classes. In Many algorithms, we used hashmap, map and list to represent the data. The process of the temporal visualization is crafted in a method called getTemporalModels. In this representation, the user can choose the year, the number of months and the start date of the data the want to visualize. If the user decide to visualize a restaurant ratings based on four months information, the data is stocked starting from the start Data entered by the user to the end date which represent an interval of 4 months. The inner workings the method getTemporalModels is shoewn below.

```
    public List<TemporalModel>
        getTemporalModels(String businessId,
        String startDate, int num_months) {
      ImmutableList.Builder<TemporalModel>
          temporalModelsBuilder =
          ImmutableList.builder();

      String[] tokens = startDate.split("-");
      int year =
          Integer.parseInt((tokens[0]));
      int month =
          Integer.parseInt(tokens[1]);

      for (int i = 0; i < 12; i++) {
         int oneStarCountTotal = 0,
             twoStarCountTotal = 0,
             threeStarCountTotal = 0,
             fourStarCountTotal = 0,
             fiveStarCountTotal = 0;
         String tickStart =
             String.valueOf(year) + "-" +
             String.format("%02d", (month -
             1) % 12 + 1);

         // accumulate star counts over
             'num_months' months and store
             them in 'result'.
         TemporalModel result;
         for (int j = 0; j < num_months;
             j++) {
            String currentDate =
                String.valueOf(year) + "-" +
                String.format("%02d", (month
                - 1) % 12 + 1);
            result =
                this.getTemporalModel(businessId
```

```
                + "_" + currentDate);

            if (result == null) {
               continue;
            }

            // accumulate star counts
            oneStarCountTotal +=
                result.getOneStarCount();
            twoStarCountTotal +=
                result.getTwoStarCount();
            threeStarCountTotal +=
                result.getThreeStarCount();
            fourStarCountTotal +=
                result.getFourStarCount();
            fiveStarCountTotal +=
                result.getFiveStarCount();

            if (month % 12 == 0) {
               year++;
            }
            month++;
         }

         temporalModelsBuilder.add(
               new TemporalModel(tickStart,
                   oneStarCountTotal,
                   twoStarCountTotal,
                   threeStarCountTotal,
                   fourStarCountTotal,
                   fiveStarCountTotal));
      }
      return temporalModelsBuilder.build();
   }
}
```

To populate the map with restaurants and their locations, we used a function called getalLocations that maps every restaurants name with its location coordinates. Since restaurants can be duplicate we used their longitudes and latitudes to situate each restaurants on the map. This method is shown is shown in the code below.

```
    public List<LocationModel>
        getAllLocations() {
      ImmutableList.Builder<LocationModel>
          locationModelBuilder =
          ImmutableList.builder();
      for (String i :
          businessIdToData.keySet()) {
         RestaurantModel restaurantModel =
             businessIdToData.get(i);
         locationModelBuilder.add(
               new LocationModel(i,
                   restaurantModel.getName(),
                   restaurantModel.getLatitude(),
                   restaurantModel.getLongitude()));
      }
      return locationModelBuilder.build();
   }
```

To keep a list of all restaurants, we have a method called getRestaurantSummary in case the list of all restaurants is needed. This method maps all restaurants with their addresses and is shown below.

```
  public List<BriefRestaurantModel>
      getRestaurantsSummary() {
      ImmutableList.Builder<BriefRestaurantModel>
          briefRestaurantModelBuilder =
          ImmutableList.builder();
      this.businessIdToData.forEach((businessId,
          model) ->
          briefRestaurantModelBuilder.add(
             new
                 BriefRestaurantModel(businessId,
                 model.getAddress(),
                 model.getName())
      ));
      return
          briefRestaurantModelBuilder.build();
   }
```

We also kept a list of all attributes and their frequency. This is done with a method called getAttributeCountMap and shown below.

```
  public Map<String, Integer>
      getAttributeCountMap(List<String>
      businessIds) {
      ImmutableList.Builder<RestaurantModel>
          restaurantModelBuilder =
          ImmutableList.builder();
      businessIds.forEach(id ->
          restaurantModelBuilder.add(this.getRestaurantM
      ImmutableList<RestaurantModel>
          attributeCountModels =
          restaurantModelBuilder.build();

      Map<String, Integer> attributeCountMap
          = Maps.newHashMap();
      attributeCountModels.forEach(model ->
          model.getAttributes().forEach(attribute
          -> {
          attributeCountMap.computeIfPresent(attribute,
              (k, v) -> ++v);
          attributeCountMap.putIfAbsent(attribute,
              1);
      }));
      return attributeCountMap;
```

To respond to the request for the heat map, we computed the pearson correlation to determine the extent to which the attribute are linearly related. This is done by generating some random numbers assign to each attribute. The code is shown below.

```
public class HeatMapService {
   public List<CorrelationModel>
       getPearsonCorrelation(List<String>
       businessId, List<String> attributes) {
      ImmutableList.Builder<CorrelationModel>
          correlationModelBuilder =
          ImmutableList.builder();
      businessId.forEach(bid ->
          attributes.forEach(attr ->
```

```
            correlationModelBuilder.add(new
                CorrelationModel(bid, attr,
                new Random().nextDouble() %
                2 - 1))));
        return correlationModelBuilder.build();
    }
}
```

## IV. CONCLUSION

In this paper we identified and estimated the impact of several variables on restaurants in the state of nevada. We mainly focused on the ratings and how they are generated and what variables are favorable to positive ratings. We also worked on attributes and how they related to the business success. We investigated the correlation between common attributes among different restaurants and their ratings. Using Anova and some data science we estimated the degree of impact these attributes have on the ratings. We discovered that some attributes contributes significantly to the outlook of the restaurant while other are just complimentary. We identified the must have attributes for any restaurant that wants to succeed. Despite the centrality of some attributes, the more attributes a restaurant has the more chances it has to succeed. With this findings, restaurants can gain insight on how to improve their business.

## REFERENCES


[1] Y. Zheng, N. Venkatesh, and W.-H. Hsieh, Yelp Insights : Data Visualization, Yelp Insights. [Online]. Available: http://shirleylei.com/projects/yelp.html [Accessed: 16-Sep-2018].

[2] Yelp Dataset. [Online]. Available: https://www.yelp.com/dataset/challenge

[3] Z. Zhang, Machine Learning and Visualization with Yelp Dataset, Medium. [Online], Available: https://medium.com/@zhiwei_zhang/final-blog-642fb9c7e781 [Accessed: 12-Sep-2018].

[4] M. Tata, Data Analytics on a Yelp Data Set . [Online], Available: http://krex.k-state.edu/dspace/bitstream/handle/2097/38237/MaitreyiTata2017.pdf?sequence=1&isAllowed=y [Accessed: 12-Sep-2018]

[5] B.Hood, V. Hwang and J. King, Inferring Future Business Attention [Online], Available: https://www.yelp.com/html/pdf/YelpDatasetChallengeWinner_InferringFuture.pdf [Accessed: 02-Oct-2018]

[6] P. Chandrashekar, N. Dhanpal, and K. Ramesh, Yelp Unbounded: Visual Analytics [Online], Available: http://karthik-ramesh.com/dv/yelp_report_final.pdf [Accessed: 14-Sep-2018].

[7] M. Fan, M. Khademi, Predicting a Businesss Star in Yelp from Its Reviews Text Alone [Online], Available: https://arxiv.org/pdf/1401.0864.pdf [Accessed:02-Oct-2018]

[8] A. Raju, C. Basnage, J. Yin Predicting a Businesss Star in Yelp from Its Reviews Text Alone [Online],Available: http://vis.berkeley.edu/courses/cs294-10-fa14/wiki/images/f/f4/Datavis.pdf [Accessed: 02-Oct-2018]

[9] J. Wang, J. Zhao, Sheng Guo, and Chris North, Clustered Layout Word Cloud for User Generated Review, SIGCHI Conference. [Online], Available: https://www.yelp.com/html/pdf/YelpDatasetChallengeWinner_WordCloud.pdf [Accessed: 01-Oct-2018]

[10] W. Yang, Y. Yuan, N. Zhang, "Predicting Yelp Ratings Using User Friendship Network Information" [Online], Available: https://pdfs.semanticscholar.org/efca/0250f18603eeb67f23a5e688591921c25f6a.pdf [Accessed: 16-Sept-2018]

[11] W. Jin, J. Zhang, "Rendering Dorling Cartograms on Yelp Data" [Online], Available: urlhttp://vis.berkeley.edu/courses/cs294-10-fa14/wiki/images/archive/1/1e/20141208214005!Paper.pdf [Accessed: 03-Oct-2018]

[12] T. Jindal, Finding Local Experts from Yelp Dataset [Online], Available: https://www.ideals.illinois.edu/bitstream/handle/2142/78499/JINDAL-THESIS-2015.pdf?sequence=1&isAllowed=y [Accessed: 3-Oct-2018]