CS1951a Final Report

Vision:

Our team started this project with the goal of building a unique recommendation engine for Yelp. During the discovery phase, we realized that most of the projects that use the Yelp dataset fall into two broad categories:

1) Using ML to predict the rating of user reviews
2) Using clustering / ML to build a recommendation engine for Yelp

While our team believed that the content of the text reviews provided rich information on users and restaurants, we realized that not many of the projects utilized this content when building recommendation engines. Thus our final idea for the project was to build a recommendation engine for Yelp based on the user text reviews.

Our team faced many challenges along the way, yet we believe we have made significant progress during this idea as we will outline below.

Data:

Our dataset of reference was the dataset publicly offered for the Yelp Dataset Challenge. The dataset contains 5 json files containing information pertaining to restaurants and users (restaurant information, reviews, tips, check-ins, etc) for a total 4.1 million reviews and 144K businesses.

While there was a lot of information in the dataset, we quickly discovered that we would have to spend a considerable amount of time cleaning the data. More specifically, we had to clean “attributes” and “categories” section of the business dataset to get coherent data, as the format Yelp provides is extremely difficult to work with (different styles of quotation, brackets, etc).

We also realized that working with all the information provided was infeasible in terms of both visualization and sheer computing power. Therefore, we decided to work with a subset of the dataset:

- Restaurants located in Nevada (second most reviewed state in the dataset)
- Restaurants with > 100 reviews
- Users with > 50 reviews

We chose these specific conditions to narrow down the dataset because we believed that we needed restaurants and users with a significant amount of reviews in order to get rich information about a particular user or restaurant.

Once we figured out the data we were going to work with, we put the data into a database. Please refer to Capstone section #1 for more information about our database.

AWS EC2

Even with our limited subset of the data, we found it difficult to work locally due to memory issues. This problem was most emphasized when we tried to calculate pairwise distance matrices -- with 50,000 reviews, the distance matrix would be 50,000 X 50,000 (and actually performing computations with these matrices would take up even more memory). Because it proved difficult to work with this kind of data, we decided to use AWS's EC2 Instance to conduct our analysis via jupyter notebook. The instance we used was the following:

AMI: Ubuntu Server 16.04 LTS (HVM)
Instance type: m4.2xlarge
CPUs: 8

Memory: 32 GB (+ 20 GB from SSD)
Storage: 2 X 80 (SSD)

Working with the Data:

Now that we had our dataset, our team wanted to process each document into a vector in order to run K-means clustering on it. We will describe the preprocessing steps, clustering and dimension reduction process, and the visualizations.

Preprocessing:

We use the SnowballStemmer from the NLTK library to stem all the words found in the corpus. We make use of two functions: one will generate both the tokens and stems, while the other will generate only the tokens. Next we create a pandas Dataframe with one column containing the tokens and another column containing the tokens -- we do this because later on we will need a mapping between tokens and stems. We also get rid of unnecessary words in the corpus using a stop words list.

Tfidf Vectorization and Dimensionality Reduction:

We use TfidfVectorizer from scikit-learn in order to transform documents into vectors. We use the following parameters after fiddling around with various configurations:
Max_df = 0.5 (all terms that are in 50% or more of the documents will be disregarded)
Min_df = 100 (all terms that are in less than 100 documents will be disregarded)
Use_idf = True (We want to use idf to weight the terms)
Tokenizer = tokenize_and_stem (We use the function described above to tokenize and stem terms)

While the output results are different based on the different configurations we utilized and the different types of clustering we performed, the above configuration will

yield an output of roughly 1,000 features for each vector on the review clustering data. Since this is still a significant amount of features, we decided to perform dimensionality reduction using the TruncatedSVD function in scikit-learn (which, performed on documents, is the same as LSA). Running dimensionality reduction on this feature set with NUM_COMPONENTS = 200 will output a explained variance of around 40%. (As the NUM_COMPONENTS goes up, so does the variance explanation. So if the NUM_COMPONENTS equals the size of the feature set, the result will yield 100% explained variance).

K-Means Clustering:

We use both the MiniBatchKMeans and KMeans provided by scikit-learn to perform our K-means clustering (MiniBatchKMeans will offer a significantly faster clustering, with only slightly suboptimal results). As for the parameters, we use:
Init = k_means++ (speeds up convergence, smarter method to initialize cluster centers)
N_init = 20 (will work through 20 initializations and use the best one -- initialization of cluster centers are extremely important in K-Means clustering as we have learned in the HW clustering songs from Spotify). Once the algorithm outputs the cluster labels for each document, we use the cluster centers to find the top N terms for each cluster.

```
Top terms per cluster:
Cluster 0: wings good food fun place chicken service fries great beer
Cluster 1: sushi roll rolls place ayce good fish great fresh service
Cluster 2: food good service don place prices better restaurant like really
Cluster 3: bar room place night good drinks great food like vegas
Cluster 4: chicken good rice fried food place ordered pork sauce salad
Cluster 5: time food experience good service got table really restaurant wait
Cluster 6: place vegas best amazing food love staff las friendly service
Cluster 7: breakfast eggs good place food pancakes great coffee service toast
Cluster 8: tacos delicious shrimp good fish taco fresh cream chips salsa
Cluster 9: great food service place good atmosphere awesome amazing friendly love
Cluster 10: burger fries burgers good place cheese food great just like
Cluster 11: order minutes food service took time said place asked got
Cluster 12: just like good ordered steak place really didn food got
Cluster 13: pizza crust good place great cheese slice just like pepperoni
Cluster 14: buffet food crab good buffets legs vegas dinner line selection
```

The clustering does a good job in classifying review text into specific categories of cuisine: “wings chicken fries beer”, “pizza crust cheese slice”, etc. However, given the sheer amount of reviews and the number of vague words in the reviews (place, food) we decided the clustering might be easier if we aggregated it into restaurants and users. While we also added some custom stopwords, it did not have a significant effect on the resulting clusters.

Restaurant Clustering

Our team conducted many different forms of clustering on restaurants. Since it would take a considerable amount of paper and time to review all these methods, we will describe the method that provided the best results and briefly include other methods in the **Other Work** section below.

We found it interesting that the clustering that resulted in the best output was actually one that we tried relatively earlier on in the project and surprisingly simple: we simply aggregate all reviews for each restaurant and run the same preprocessing steps, Tf-idf vectorization, dimensionality reduction, and K-means clustering as above. If the previous process resulted in each review text being put into a cluster, this process would result in each restaurant being put into a cluster (hence restaurant clustering).

The resulting clusters demonstrate a very good categorization of restaurants into specific cuisines, but contain *much richer information* than what can be found in typical cuisine groupings:

```
Top 10 terms per cluster:
Cluster 0: crab shrimp lobster seafood oysters boba cajun fish tea fries
Cluster 1: burger fries burgers bacon shake bun patty fast double cheeseburger
Cluster 2: bar beer fries wings burger hour beers dog sandwich server
Cluster 3: wine dish pasta bread italian dessert lobster chocolate course chef
Cluster 4: steak steakhouse lobster filet steaks rib prime bread potatoes bar
Cluster 5: mexican salsa chips tacos burrito beans guacamole taco margarita nachos
Cluster 6: thai pad curry rice noodles spicy soup tom dish beef
Cluster 7: buffet buffets station legs crab curry breakfast selection 99 sushi
Cluster 8: sushi ayce roll rolls nigiri fish tuna tempura salmon sashimi
Cluster 9: chinese rice soup noodles pork sum beef dim noodle shrimp
Cluster 10: pizza crust pizzas pepperoni toppings delivery slice garlic wings pie
Cluster 11: italian pasta bread olive lasagna garden alfredo dish wine meatballs
Cluster 12: pita hummus greek mediterranean lamb rice fries bread beef feta
Cluster 13: breakfast eggs pancakes hash toast cafe steak coffee omelet egg
Cluster 14: pho vietnamese broth rolls pork noodles rice spring egg bowl
Cluster 15: coffee chocolate cake pastries bakery cream nutella breakfast tea ice
Cluster 16: tacos taco asada carne burrito mexican salsa beans rice burritos
Cluster 17: sushi rice ramen poke japanese miso roll bowl tuna spicy
Cluster 18: sandwich sandwiches deli bread soup smoothie turkey healthy soups salads
Cluster 19: bbq korean brisket ribs pork mac hawaiian meats sides rice
```

For example, observe Cluster #19. This may be a surprising cluster because “korean” is grouped together with other meat keywords: bbq, brisket, ribs, pork. However, this was an even more surprising result because we know (all four of us being from Korea) that while typical Korean food is definitely not limited to meats, the majority of Korean restaurants in the United States are centered around bbq-ish themes (unlimited bbq, etc).

We also note that all of the other clusters capture a specific category as well: some notable ones are the Dessert Cluster (coffee, chocolate, cake, pastries, bakery…), Japanese Cluster (sushi, rice, ramen, japenese…), Japanese Cluster 2 (more focused on rolls - sushi, roll, rolls, nigiri, fish, tuna). This method also generates the best clustering visualization (displayed below in the **Visualization** section).

Let's compare this to Yelp's categorization. For Japanese restaurants, Yelp only offers two categories namely "Japanese" and "Sushi Bar" while, as seen from our cluster, Japanese cuisine boasts varieties of foods ranging from "ramen" to "tempura" to "sashimi" to "sushi". Such phenomenon can be observed across all cuisines. Since these results seem promising, we attempt to recreate the same process for clustering users.

User Clustering:

Here, we realize that user clustering with the review aggregation method does not work. The reason is because there is a huge difference between aggregating reviews for each restaurant (which will capture attributes related to the restaurant) and aggregating reviews for each user (which will capture attributes related to the user themselves, not the restaurants they go to):

```
Top 10 terms per cluster:
Cluster 0: pros cons fav nonetheless ambiance okay typically bf despite honestly
Cluster 1: okay girl terrible fantastic ambiance latte awful forward groupon return
Cluster 2: challenge 100 husband boba okay filipino 99 milk pho donuts
Cluster 3: lol haha boyfriend okay ayce ramen pho boba alright 95
Cluster 4: husband yummy okay yum kids return wonderful absolutely dog lol
Cluster 5: fantastic ambiance ramen yummy absolutely return korean bartender japanese wonderful
Cluster 6: cause boyfriend yummy lol kinda seriously totally bartender greeted checked
Cluster 7: pho ramen ayce okay korean solid broth 95 99 nigiri
Cluster 8: com select ramen ayce foie gras 95 brunch pics nigiri
Cluster 9: kids daughter yummy fantastic son husband return lol wonderful 00
Cluster 10: wife fantastic value okay outstanding ramen kids return burrito korean
Cluster 11: yummy lol filet ambience completely mgm absolutely pretzel caesar dj
Cluster 12: girlfriend pho korean okay tofu shabu boba pad broth anyways
Cluster 13: yummy lol yum vegan soooo totally boyfriend dog sooo helpful
Cluster 14: okay solid ramen nigiri fantastic ambience pho absolutely benedict noodle
Cluster 15: boyfriend ramen boba pho okay ayce broth yummy cute milk
Cluster 16: bf boyfriend okay yummy kinda ayce yum lol ramen nigiri
Cluster 17: okay kinda greeted damn hostess dog absolutely nachos return minute
Cluster 18: alright chicago ranch crew kinda dog cilantro boba pepperoni solid
Cluster 19: dont thats pho bucks lol absolutely wife wonderful guys korean
```

We see a lot of repeating terms: husband, boyfriend, wife, kids, daughter, yummy, etc. While not a particularly successful clustering, it definitely is interesting to see what kind of traits the clustering captures. While we did not do so for this project, our team believe that it would be interesting to delve deeper: with the correct stop words, parameters, and different clustering methods, would it be possible to cluster users based on what they believe to be important (ambience, family members, etc).

Metrics

$$User - based: \sqrt{\sum_{i=1}^{user} \sum_{c=1}^{res_k} (r_{i,c} - \mu_{i,c})^2}$$

$$Cluster - based: \sqrt{\sum_{j=1}^{user_k} \sum_{c=1}^{res_k} (r_{i,c} - \mu_{j,c})^2}$$

$$Restaurant - based: \sqrt{\sum_{l=1}^{r} \sum_{c=1}^{k} (u_{l,c} - \mu_{l,c})^2}$$

First, to measure our user and restaurant clustering based on reviews, we tried using traditional measures of RMSE (Root Mean Squared Error). To be more specific, in the restaurant cluster case, with each user, we measure the mean of the ratings of the restaurants in a cluster. Then we subtract each rating from the mean, square, and sum the error. While such error metric is common in the literature, since we didn't account for ratings directly, our result didn't fare well compared to the baseline although we initially assumed that reviews will latently account for ratings. While the user clustering was slightly better, the difference wasn't significant. After lengthy discussion, we realized that the particular subset of restaurants and users we chose (restaurants > 100 reviews, users > 50 reviews) might explain such little difference between our cluster and the baseline of random cluster.

We decided that, there wasn't really good, universal way to measure the cohesiveness of our restaurant / user clusters without relying on ratings. So instead we tried a lot of different proxy or interesting ways to measure and visualize our clusters, which follows below. One method was to use to Yelp's categorization. If our restaurant cluster was indeed clustered well, as seen from top 5 terms, our clusters would have shorter set of categories compared to random clusters. So if we add all the categories of restaurants in the cluster, we can compare the length of set of categories between clusters. This time, our clusters indeed fared better with shorter length sets.

Visualization

We spent a lot of effort in generating visualizations since we found it difficult to create a suitable metric to measure our clustering results. This section will outline our methodology, display some of our visualizations, and contain our thoughts about the process.

Methodology:

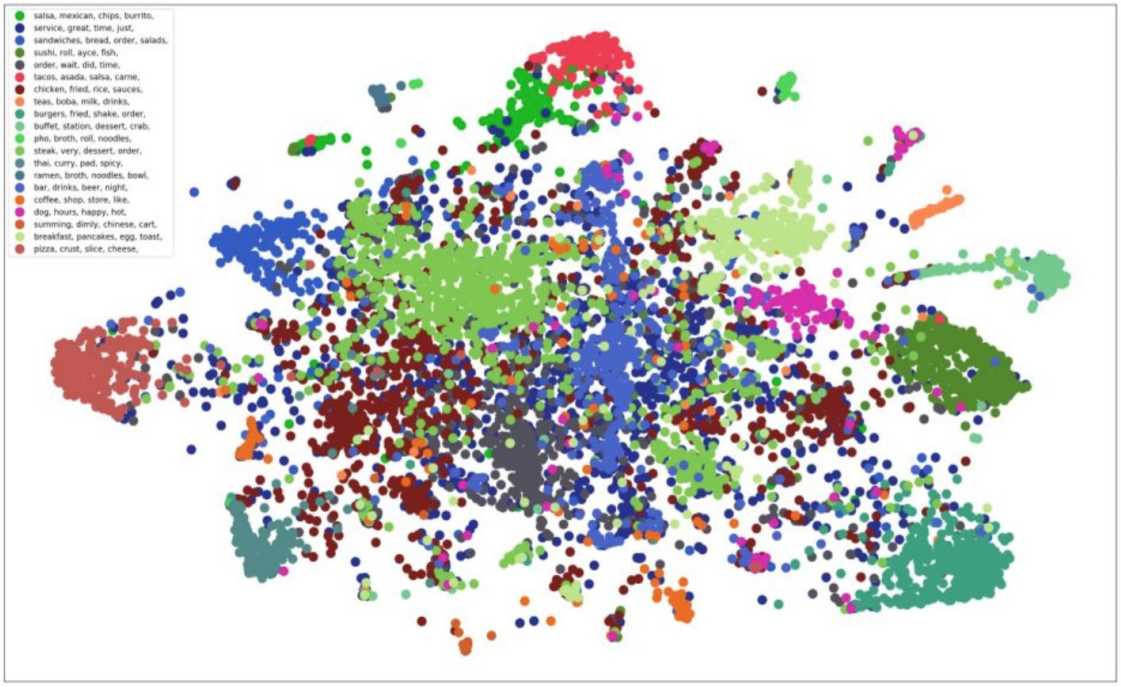
Since the vectors we are working with are still extremely high dimensional (relative to what we can plot), our team needed to figure out what the best method would be to generate various visualizations. A brief overview of our methodology:

- Create a pairwise distance matrix using the vectors we created using Tf-idf and dimension reduction (LSA) using scikit-learn's pairwise distance functions (cosine_similarity, euclidean_distances).
- Use this distance matrix to feed as input into various dimension reduction methods; we used Multidimensional scaling (MDS) and t-SNE.

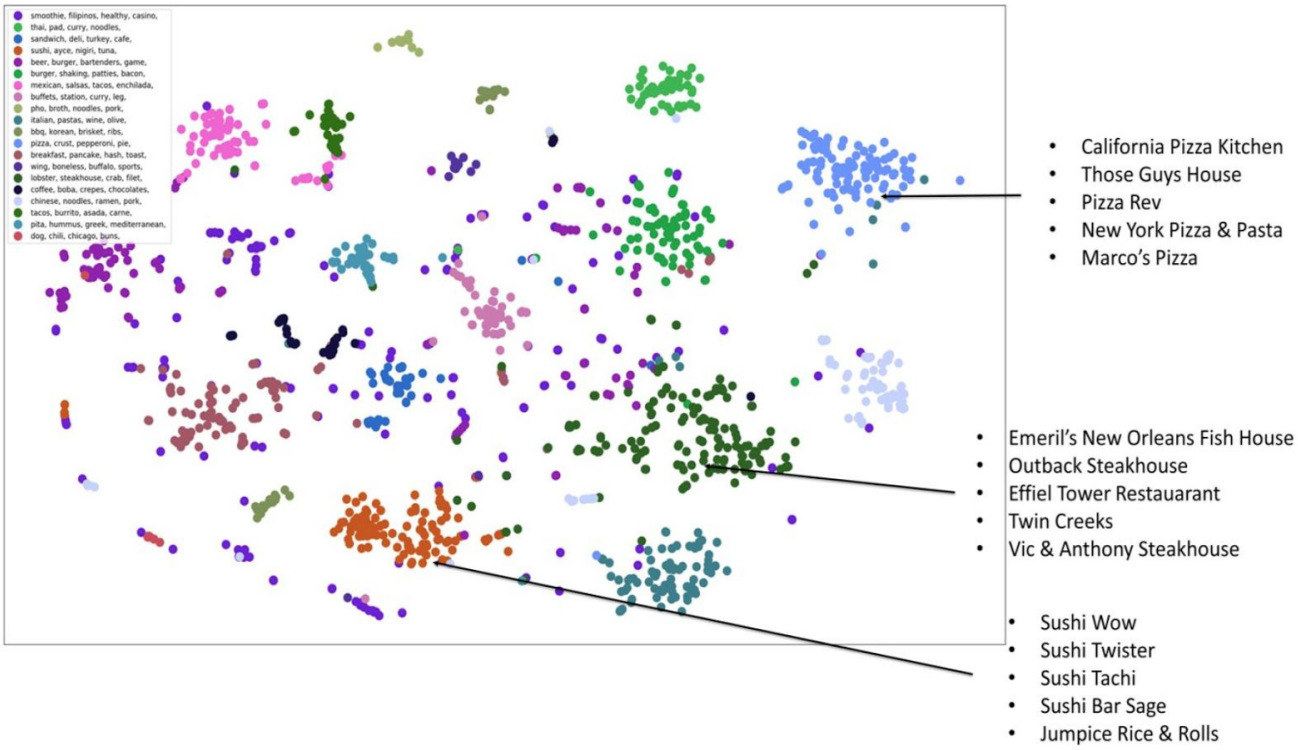
These steps will output coordinates for each point (restaurant, review, etc) and we use these coordinates to create visualizations in various forms: we use plot.ly, matplotlib, and d3 to create custom visualizations. We test around with various configurations (in both the clustering process and also in the visualization process itself).

Visualizations:

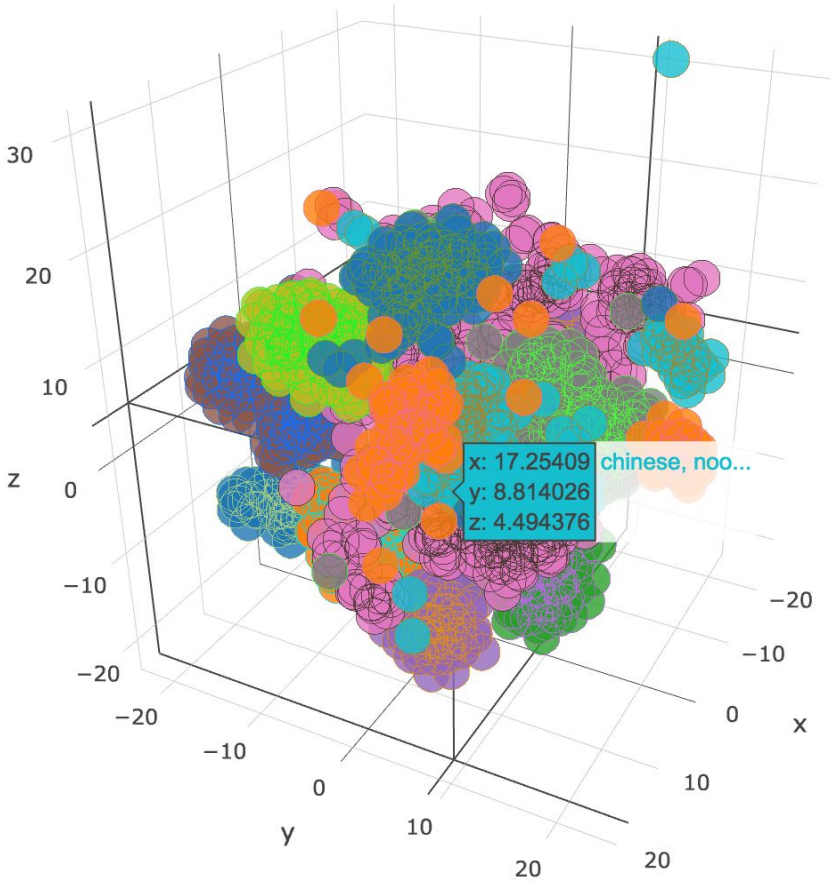
1. Visualization of K-means clustering on review text using the process described above, using dist = 1 - cosine similarity and t-sne dimensionality reduction. Each dot represents a review text and each color represents a cluster. Plotted with matplotlib.



2. Visualization of K-mean clustering on restaurants using review aggregation, using dist = 1- cosine similarity and t-sne dimensionality reduction. Each dot represents restaurant and each color represents a cluster. Plotted with matplotlib.



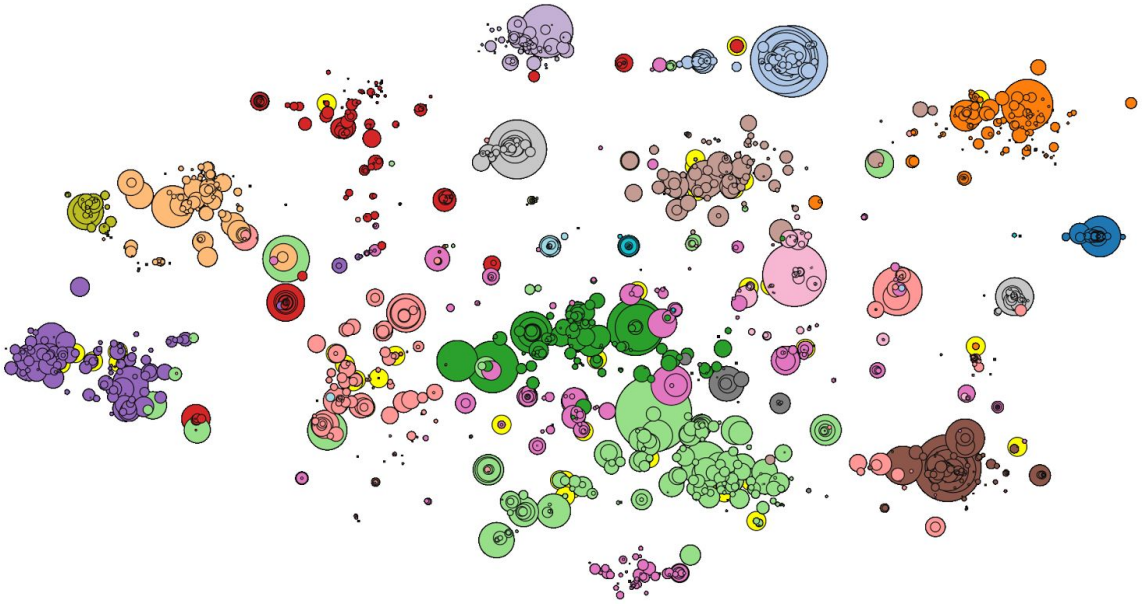
3. Visualization of K-mean clustering on restaurants using review aggregation, using dist = 1- cosine similarity and t-sne dimensionality reduction to **3 dimensions**. Each dot represents restaurant and each color represents a cluster. Plotted with plot.ly (interactive 3d plot will be included in the handin)



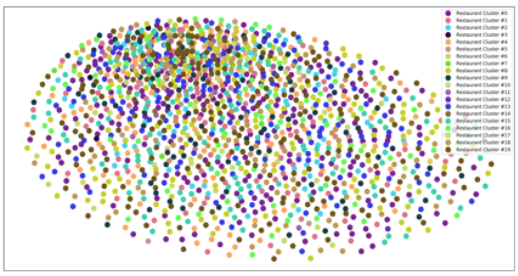
4. Visualization with D3. The size of dots is proportional to the number of review texts for restaurant. When user id is queried, restaurants that user wrote review changes blinking yellow dots.

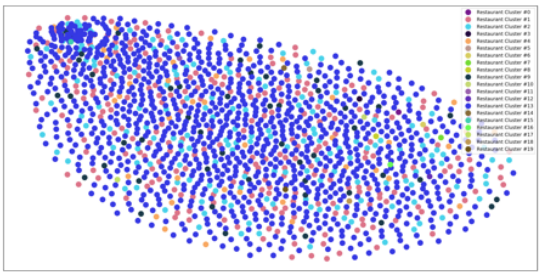
User Id: QHX3NxFAlda3l32taYzl8

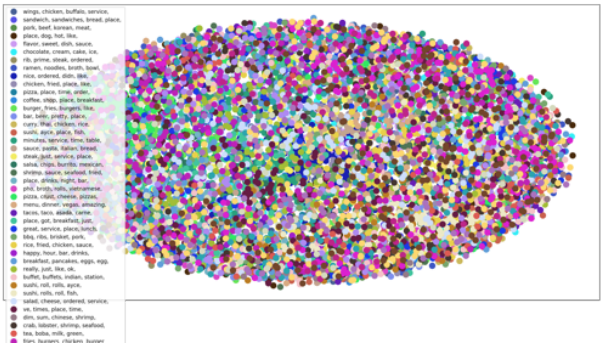
Find Restaurants!



5. Failed Visualizations / Clusterings (Euclidean Distance / MDS dimension reduction)







Visualization Conclusion:

We generated many, many more visualizations throughout our project (different dimension reduction techniques, number of clusters, Tf-idf parameters, K-means clustering parameters, etc). We will include a subset of these visualizations that we found to be the most interesting in the hand-in, including the interactive 3d visualization generated with plot.ly and the custom made D3 visualization.

Having tried many visualizations and methods of clustering, the following are our observations

- t-SNE > MDS for our dataset
- Euclidean distances don't work in high-dimensional spaces (converges to 0)
- K = 20 provides a good clustering of restaurants (observing both the visualizations and top terms per cluster)
- It's important to figure out which visualizations will be the most useful for the user -- in our case, we believed it would be most useful for a user to be able to navigate around the clusters and observe the cluster names and restaurant information, as well as figure out what restaurants a specific user left reviews for in the context of the clustered restaurants.

Recommendation

Please refer to **Capstone Section #2** for information about our recommendation system.

Note on ML / Stats:

While we don't have a separate area of ML / Stats, we believe we fulfilled our requirements for ML / Statistical analysis from our discussion above:

1) K-means clustering
2) Dimensionality reduction for clustering (LSA)
3) Dimensionality reduction for visualization (t-SNE, MDS)

From our work on the project (in the blog posts but not discussed above):

1) Extracting top 10 one-star and five-star review tokens using Multinomial Naive Bayes classifier
2) Using a classifier to predict the ratings from text reviews (Linear SVM), including the positivity / negativity of a review
3) LDA (topic modeling) to discover topics in our corpus

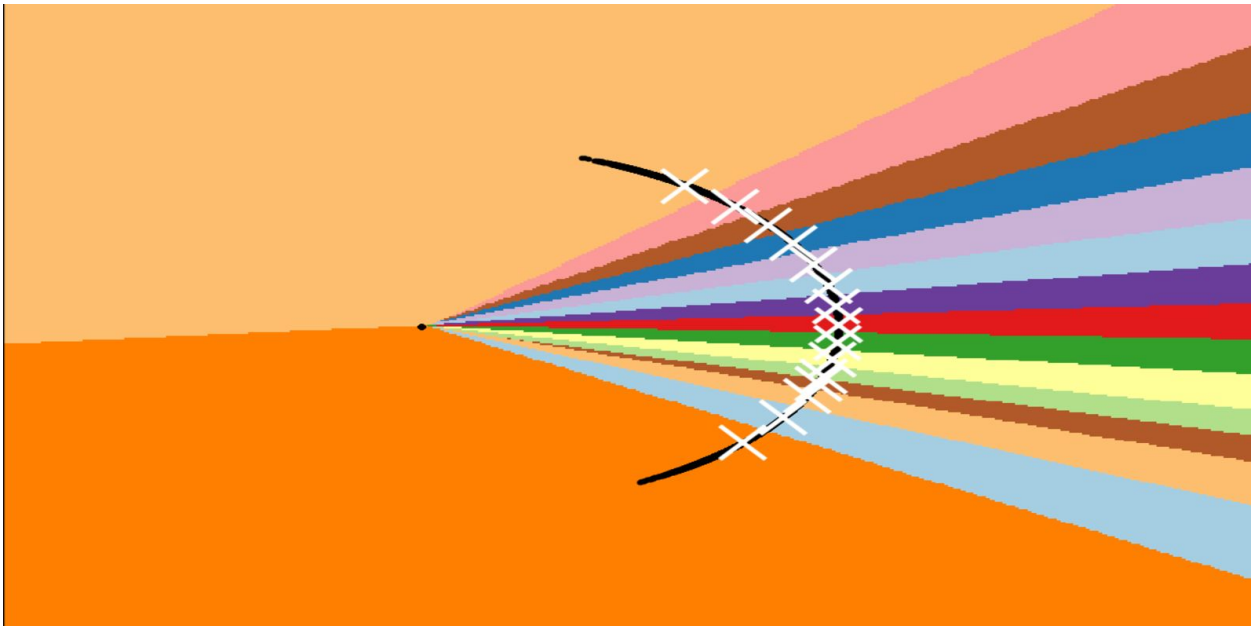
We believe these also fall under the realm of ML / Stats and the blog contains more information on them -- while they were informative during the discovery phase of

our project, we do not discuss them in depth because they do not have a great effect on the outcome of our final project / vision.

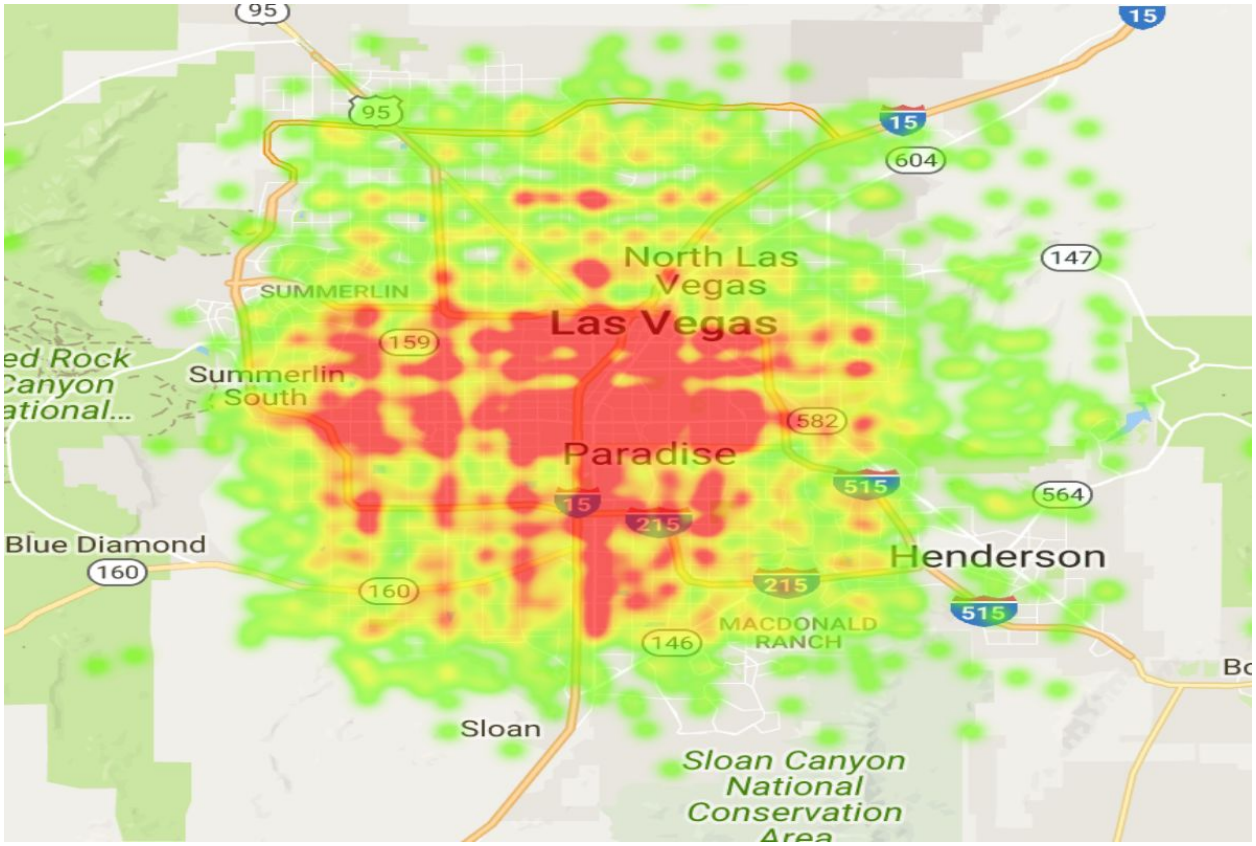
Other Work

Some of the “side projects” we did were mentioned above, but here are some additional ones:

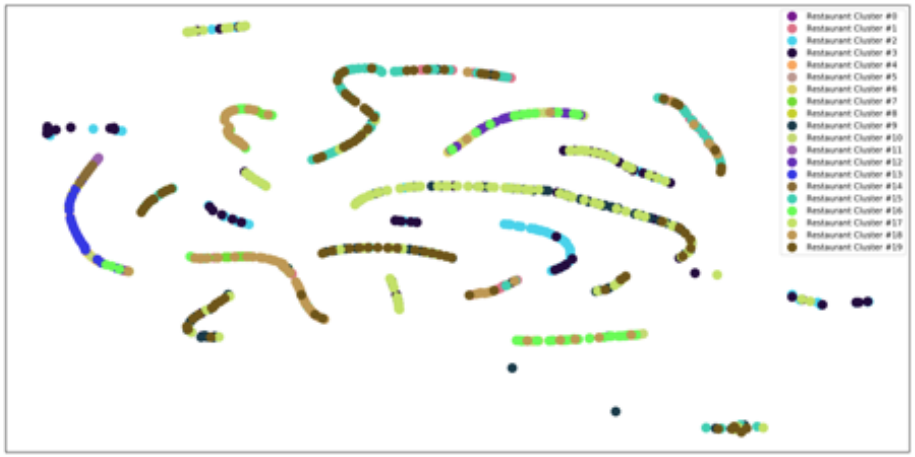
1) Visualization after reducing the dimension to 2 dimensions *before* running K-means clustering (whereas in the final version we reduce the dimension to ~200 dimensions before running clustering, then reduce the distance matrix to 2 dimensions):



2) Map Visualization of Restaurants in Las Vegas (other visualizations pertaining to the dataset can be found in the blog posts)



3) Clustering restaurants based on the features [average rating, number of ratings]:



Capstone Information

This project is used as capstone for one of our group members, Dae Hyun Kim.

For the web app recommendation system & data, please refer to the github.
https://github.com/junsu10291/CS1951ABlog

#1 Use of a database:
Of all the database we have investigated, we chose SQLite as our database. SQLite, which is installed on all department machines, was more than enough to meet the requirement of our final project.

After preprocessing data from the original Yelp datasets (~3.5GB), 'yelp_academic_dataset_review.json' and 'yelp_academic_dataset_business.json', we stored our data (517MB) in sqlite3 database called yelp.db.

.schema =
CREATE TABLE review(
id integer primary key, user_id text, text text, business_id text, stars real, date text);
CREATE TABLE business(
business_id text primary key, address text, categories text, city text, review_count integer, name text, longitude real, stars real, latitude real, postal_code integer);

Schema of Table 1 : review = (id: integer (primary key) , user_id: text, text: text, business_id: text, stars: real, date: text)
There are total of 635149 rows and 6 columns (# of features) of data in review table.

Schema of Table 2 : business = (business_id: text (primary key), address: text, categories: text, city: text, review_count: integer, name: text, longitude: real, stars: real, latitude: real, postal_code: integer)
There are total of 1624 rows and 10 columns (# of features) of data in business table.

After we make a secure connection between python and our database using sqlite3.connect function, we query data from the database to a pandas dataframe using pd.read_sql_query function.

#2 Build a data pipeline:
We devised two different methods for recommending restaurants to users, one using restaurant pre-clustered data and another using user pre-clustered data.

The first recommendation system gives recommendations based on a user's choice of keywords. It compares the input keywords (separated by a space) with the lists of top terms associated with each restaurant cluster and predicts which existing restaurant cluster would the keyword be classified as. To calculate the semantic similarity between two words, it uses NLTK.wordnet, a database of English words that are linked together by their semantic relationships. The more two words are related, the closer to 1 the wordnet value is. And, two non-related words have a wordnet value close to 0. Finally, it simply ranks the list of restaurants in the classified cluster by their average-star ratings in descending order. If wordnet does not recognize the user's input keywords, it will output error and give general recommendations(shuffled restaurants with 4-5 average star ratings).

The second recommendation system gives personalized recommendations based on looking at other users with similar tastes. It is required to enter an existing USER_ID in our database for this recommendation system as it is geared towards giving tailored suggestions for each and every unique user. First, the system finds a specific user cluster that a user with USER_ID belongs to. Then it iterates through all the users in the user cluster and accumulates the count of how many times a restaurant has received 5 star reviews from all these users. Then it ranks by the count in descending order and outputs the list of restaurant recommendations. If non-existing USER_ID is given as input, it will output error and give general recommendations (shuffled restaurants with 4-5 average star ratings).

In the future, we can improve our second recommendation system by adding a feature where we can add a new user with his or her preferences to our database and instantly classify the new user into one of the pre-existing user clusters and give recommendations accordingly.

We used pre-clustered data because it takes too long to run a k-means clustering on a large set of data. It also does not make sense to train a classifier every time we search for something new. We probably would update the model once a day or even once a week. The pre-clustered data was exported ahead of time by running intensive computations on EC2 instance (m3.2xlarge, ~30 GB RAM, +20GB allocated SSD) on Amazon AWS.

#3 Front-end:

We built a user interactive web application that recommends restaurants to users solely based on a large number of review texts sampled from Yelp Datasets. We chose Flask,

a python web micro framework, because we can run python scripts directly from Flask. Installing codes for Flask is included in the python script.

Our recommendation system gives a list of recommended restaurants based on a user's inputs, keywords or USER_ID. To double the interactivity, we included dynamic data visualizations of the queried data using d3 and plotly libraries.

Challenges & Conclusion

Our team faced many difficulties along the way. For example:

1) Visualizing clusters in high dimensional spaces / making sense of the resulting clusters
2) Coming up with a universal metric to measure the outcome of a restaurant cluster or a user cluster was difficult because we had several different attempts for clustering (using numerical features, text features, and more).
3) We failed to come up with a reasonable user cluster based on text reviews; however, we gained important insight and believe further work could be done in this area

Despite these challenges, we managed to implement a unique, free-form text based recommendation system that is built on text reviews from the Yelp dataset. As seen in the live-demo from the presentation, which you can personally check out in the github, the generated recommendations do a pretty good job of suggesting similar restaurants with high ratings. We have also tried our best to mitigate the challenges we have faced -- mainly the lack of a metric -- by focusing more on other types of analysis (many forms of visualization). All of our team members have learned a lot during this project (pandas Dataframes, jupyter-notebook, AWS EC2 Instances, scikit-learn, visualizations, clustering, etc) and have made use of the skills learned during the semester (databases, D3.js, data integration, K-means clustering, ML classifiers). Given more time in the future, we would love to work further on this project by developing more visualizations, refining the recommendation system, and delving deeper into user clustering.