NLP analysis & Recommendation system for Yelp

University of California, Los Angeles
March 12, 2020
Jiancong Sun, Yifeng Lan, Shaohua Xiao, Ruodi Huang

Overview

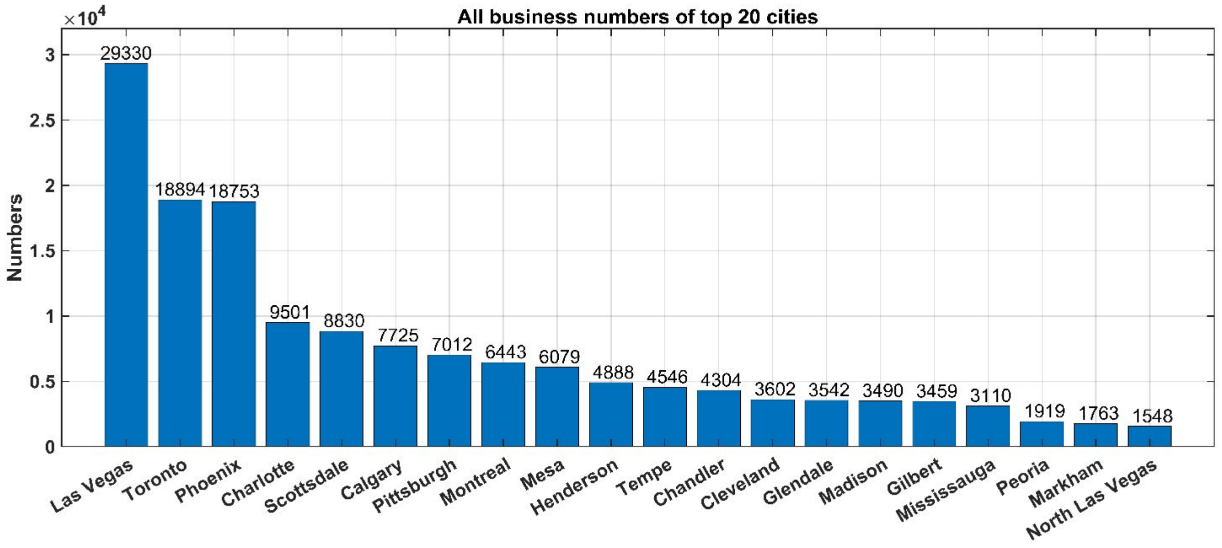
This dataset is a subset of Yelp's businesses, reviews, and user data. It was originally put together for the Yelp Dataset Challenge which is a chance for students to conduct research or analysis on Yelp's data and share their discoveries. In the dataset we can find information about businesses across 11 metropolitan areas in four countries.

EDA and Visualization

Before the data cleaning, there are 6,685,900 reviews given by 1,637,138 users to 192,609 businesses. The cleaned dataset consists of 6,661,896 reviews for 192,437 businesses.

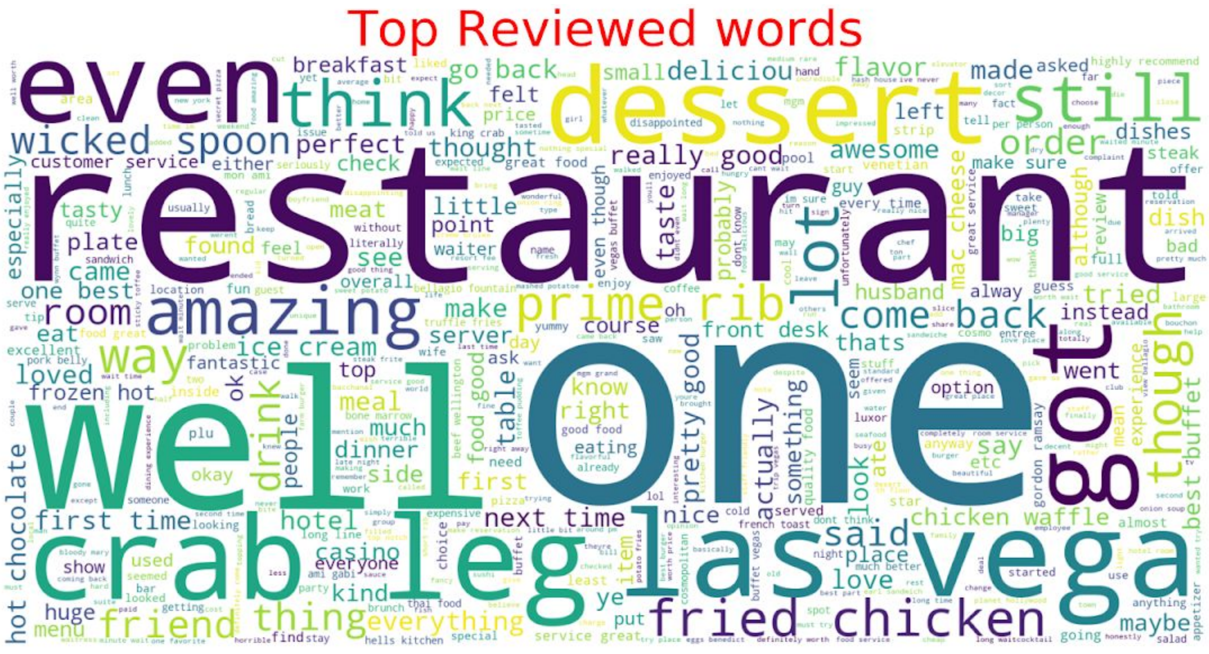
Users and reviews: Most users give 2-4 reviews and the average review star is 3.72..

Business: There are 59106 different types/categories of Businesses in Yelp. The top three popular categories are restaurants, shopping and food.



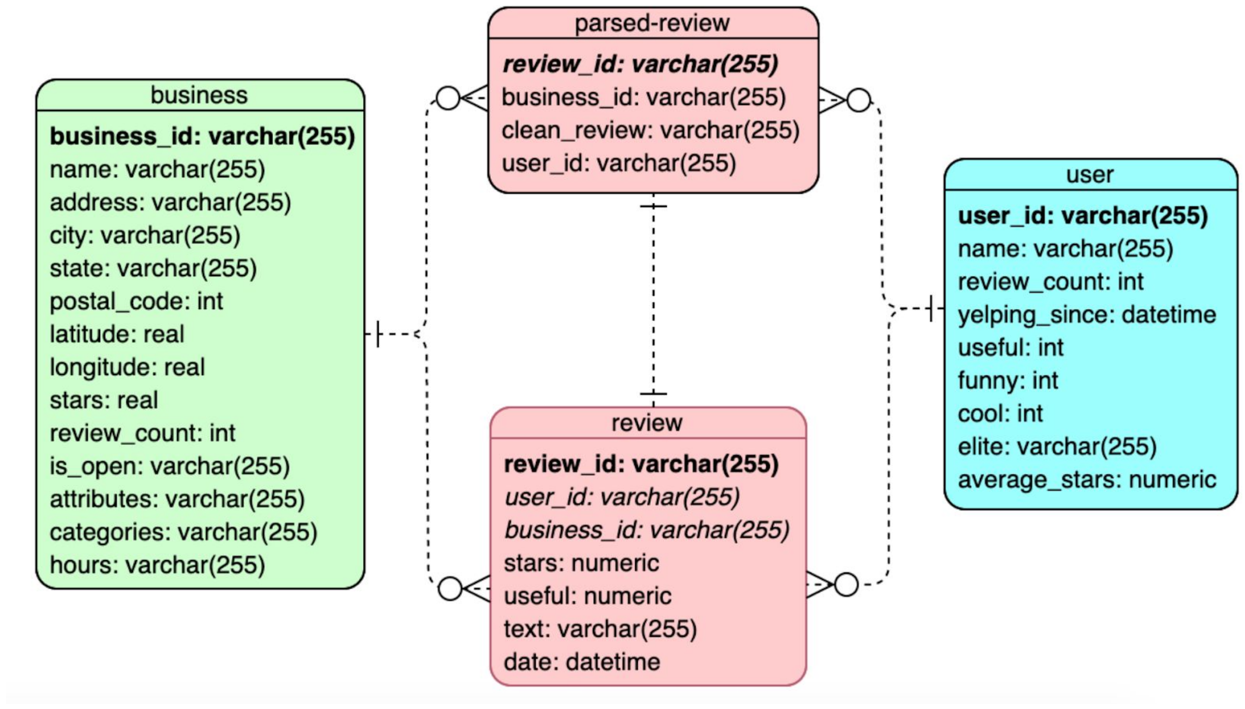
Cities:The dataset includes information span over 10 metropolitan areas. The top three popular cities are Las Vegas, Phoenix and Toronto. Therefore, we will focus on restaurants in Las Vegas.

Las Vegas: There are 2014024 reviews for 29330 businesses in Las Vegas. There are 8581972 reviews for restaurants in Las Vegas. A glimpse of the 'wordcloud' leads to the general observation that the Las Vegas food largely consists of bars, and seafood. Crab leg, prime rib, dessert and fried chicken.



Data Preparation for Modeling

The original dataset is a combination of multiple json files. In order to read the data easier and faster, I created a database and insert all data inside.



With the review text data, we use spaCy and gensim to do the text processing. SpaCy is a good library for breaking down the paragraphs into words, and performing changes such as text normalization and removing stop words. Given a full text data, the first step is to create unigram, bigram and trigram models. Unigram did the normalization job, and both bigram and trigram models connect frequent words together. Such as New York, the model can understand that it's a phase New_York.

After that, Latent Dirichlet allocation (LDA) model is used to group the words into topics. I tried 10,30 and 50 topics and found 30 had better accuracy. These 30 topics are related to different types of food, environment, services, etc. A given text string can be broken down to something like (topic 1: 0.5, topic 2: 0.3, topic3: 0.1 ...) We will use the result in text vectorization and content based recommendation model.

Word2Vec

The goal of word embedding models is to learn dense numerical vectors to represent each term in a corpus of vocabulary. With word embedding, the vectors will learn about the meaning of the terms and the relationships between terms in the vocabulary. By learning the meanings and relationships independently from any previous background knowledge, word vector models is an unsupervised model.

Among all the word embedding models, word2vec is a very popular one, which combines the information given by the words immediately before and after the target words to interpret the meaning of the target words.
Word2vec has the following three user-defined hyperparameters:

1. The dimensionality of the vectors. Typical choices include a few dozen to several hundred.
2. The width of the sliding window, in tokens. Five is a common default choice, but narrower and wider windows are possible.
3. The number of training epochs.

We trained word2vec on trigram_sentences_all.txt, which has been cleaned by Eric, using 100-dimensional vectors and setting up our training process to run for twelve epochs.

Having learned a quantitative vector representation for each term, we created a pandas DataFrame with the number of rows equal to the number of terms in the vocabulary — and 100 columns as the 100 dimensions of word2vec model.

To apply this DataFrame, we could look up related words and phrases for some given interested terms.

Recommendation Models

Location Based Model

The location based model uses K-means clustering to group the restaurants in Las Vegas by their longitudes and latitudes. It then ranks the restaurants in each cluster by star-rating. To make a recommendation, it first identifies the longitude and latitude of interest to the user and assigns a restaurant cluster to the user based on location. It provides the user with the top-ranked restaurants in the cluster.

The Kmeans function in the sklearn.cluster library was used to fit a model. To determine the number of clusters to use in the Kmeans model, an “elbow plot” was used, which indicated that k=5 is a reasonable number of clusters. The silhouette scores for a series of k, ranging from 2 to 50, were also calculated.

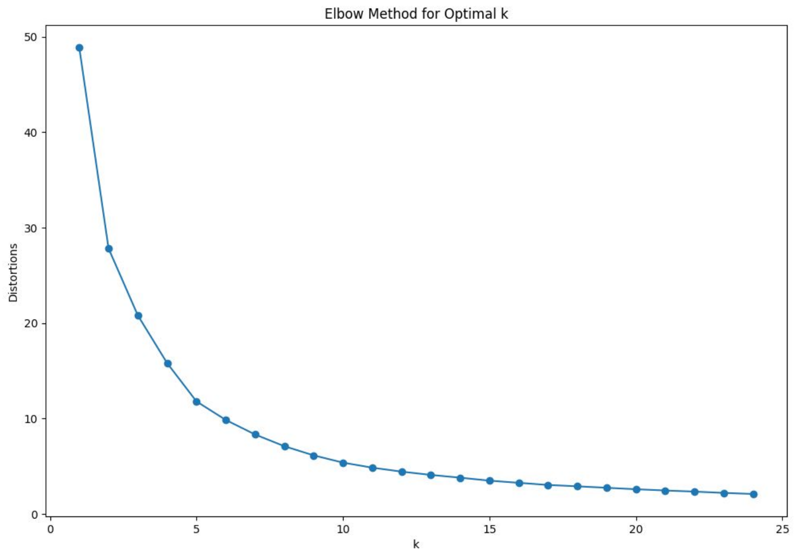
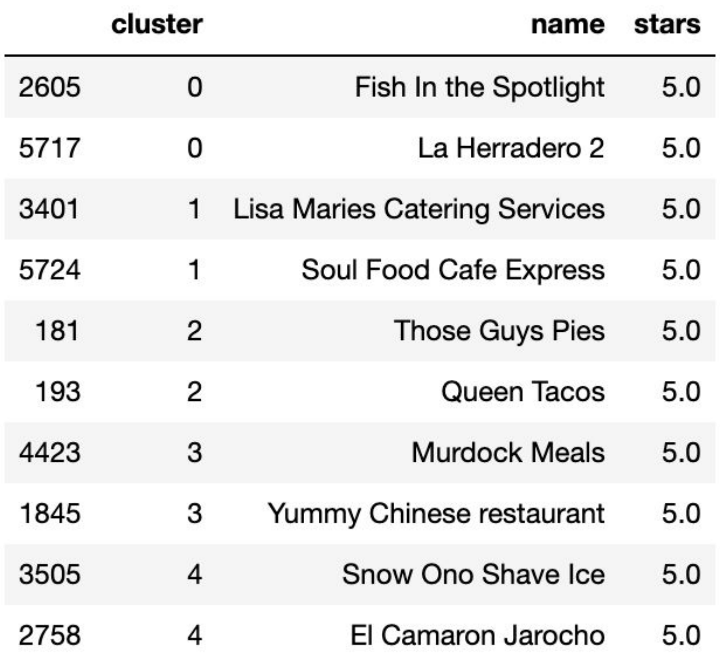


Figure. Elbow plot to determine the number of clusters.



	cluster	name	stars
2605	0	Fish In the Spotlight	5.0
5717	0	La Herradero 2	5.0
3401	1	Lisa Maries Catering Services	5.0
5724	1	Soul Food Cafe Express	5.0
181	2	Those Guys Pies	5.0
193	2	Queen Tacos	5.0
4423	3	Murdock Meals	5.0
1845	3	Yummy Chinese restaurant	5.0
3505	4	Snow Ono Shave Ice	5.0
2758	4	El Camaron Jarocho	5.0

Figure. Top 2 ranking restaurants in each location cluster.

Content Based Model

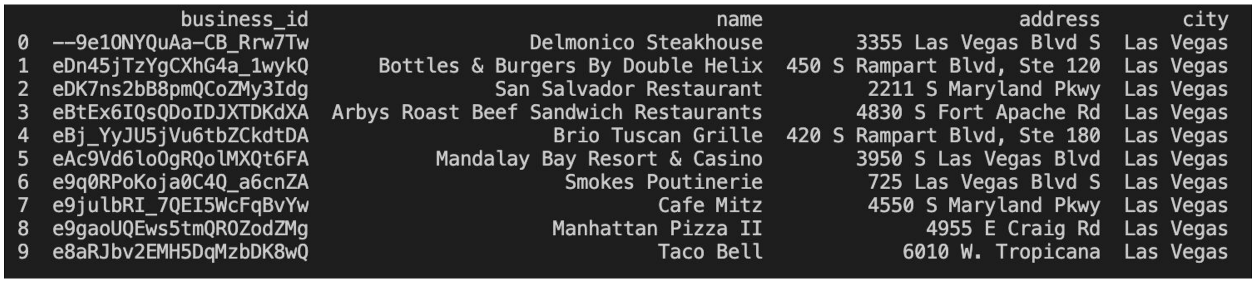
The content based recommendation is centered around finding the similarities between the test set, restaurants not reviewed by a user, and training set, the restaurants that the user has reviewed. Based on how the user ranks the previous restaurants, the model predicts whether the user will like the new restaurant and makes recommendations.

The attributes of restaurants and users were scaled to between 0 and 1. Features of restaurants and all the ratings that the user has given are multiplied to generate a user profile/model. Cosine similarity was used to assess the relationship between new restaurants and the user profile. Restaurants with high similarity ranking were recommended.

We also included an option to add location constraint to the recommendation, i.e. highly recommended restaurants within a certain radius to the location of the user.

Recommendation Example: for the user posted the below reviews previously, our system will recommend the restaurants in the following graph:

> "...fast service seat fact different kind french_fry chicken strip wander place fairly_inexpensive burger great evening Penn_Teller..."
> "...think good fettuccine_Alfredo life sauce buttery creamy excellent day..."
> "...wow margarita nachos steak excellent..."
> "...personally opt combo plate taco tostada good chance..."



```
              business_id                                  name                    address       city
0  --9e1ONYQuAa-CB_Rrw7Tw                   Delmonico Steakhouse       3355 Las Vegas Blvd S  Las Vegas
1  eDn45jTzYgCXhG4a_1wykQ      Bottles & Burgers By Double Helix  450 S Rampart Blvd, Ste 120  Las Vegas
2  eDK7ns2bB8pmQCoZMy3Idg              San Salvador Restaurant        2211 S Maryland Pkwy  Las Vegas
3  eBtEx6IQsQDoIDJXTDKdXA  Arbys Roast Beef Sandwich Restaurants       4830 S Fort Apache Rd  Las Vegas
4  eBj_YyJU5jVu6tbZCkdtDA                    Brio Tuscan Grille  420 S Rampart Blvd, Ste 180  Las Vegas
5  eAc9Vd6loOgRQolMXQt6FA           Mandalay Bay Resort & Casino        3950 S Las Vegas Blvd  Las Vegas
6  e9q0RPoKoja0C4Q_a6cnZA                    Smokes Poutinerie        725 Las Vegas Blvd S  Las Vegas
7  e9julbRI_7QEI5WcFqBvYw                            Cafe Mitz        4550 S Maryland Pkwy  Las Vegas
8  e9gaoUQEws5tmQROZodZMg                   Manhattan Pizza II            4955 E Craig Rd  Las Vegas
9  e8aRJbv2EMH5DqMzbDK8wQ                            Taco Bell           6010 W. Tropicana  Las Vegas
```

Collaborative Filtering

Collaborative filtering recommendation takes in user ratings for different restaurants, compare them with the ratings from the other users. The model finds similar users and uses the ratings from these users to predict whether the given user will like or dislike a specific restaurant.

I use python library Surprise to find the best clustering model based on RMSE, MAE and fitted time. According to the table below, we see that SVD and SVD++ models have better performance. They have similar performance but the fitted time for SVD++ is way more than the SVD model. Therefore, SVD is my final model for collaborative filtering.

Model Selection:

	test_rmse	**test_mae**	**fit_time**	**test_time**
KNN Basic	1.0306	0.7968	0.27	7.35
KNN Baseline	0.9834	0.7587	0.86	8.89
KNN WIth Means	0.9913	0.7655	0.40	9.69
SVD	**0.9633**	**0.7510**	**23.65**	1.90
SVDpp	**0.9646**	**0.7499**	**748.89**	31.52
SlopeOne	0.9998	0.7701	7.60	21.41
NMF	1.0340	0.8014	22.66	1.40

With SVD model, I use GridSearchCV to find the best parameter for this data:
(n_epochs=25,lr_all=0.01,reg_all=0.4)

Recommendation System Design

Given different use cases and scenarios, this recommendation system is designed to return the most applicable results to our users. The models above are taking upto two parameters: user_id and address (optional). And for a user_id, it can be a new user (less 10 reviews) or a frequent user (more than 10 reviews). As a result, we have four different use cases here:

	without address	**with address**
new users	top restaurants in the City	location_based
frequent users	collaborative filtering content_based	collaborative filtering w/ distance content_based w/ distance

This design can maximize the utilization of available user information to make precise restaurant recommendations. In order to give our user more options, the recommendation system will return different results when the user refreshes / calls the API again.

In the future, the model should include 75% fitted recommendation and 25% various options.

Conclusion

In this project we used natural language processing and word embedding algorithms to process text information in the reviews in the Yelp dataset. As a result, a large amount of user and restaurant information can be extracted to develop models to provide recommendations to users and business. Focusing on restaurants in Las Vegas, we developed several recommendation models, including location-based, content-based, and collaborative filtering models, to provide suggestions to users. In the future, we would like to further develop the recommendation system to include more data and improve speed, and incorporate it into UI and APIs.