ECE656 Project – Yelp Dataset

Tiankai Jiang

March 15, 2020

1 Introduction

Yelp dataset is a subset of Yelp's review, business and user data. The Yelp dataset used in this project is 2019 version, which contains 6,685,900 reviews, 192,609 businesses and 1,637,138 users from 10 metropolitan areas. Details of this dataset can be found here.

2 Data Preprocessing

2.1 Data Preview

We can easily check that all user_ids in tip.json and review.json have records in user.json. But lots of user_ids in friends are missing. And we can also check that all business_ids in tips.json, review.json, checkin.json and photo.json are in business.json. And we can verify that the relationship between friends are mutual, that is, if A appears in B's friend list, then B will appear in A's friend list.

Also, we get the following information: all ids are 22 characters long; maximum username length is 32; maximum business name length is 64; maximum review length is 5000 and maximum tip length is 500.

The most complex part is the attributes and categories in business information. There are 1300 different categories and 39 different attributes for business. In those attributes, 32 of them have a single value, e.g. "True", "False", "None". The rest of them contain nested structure, which means their value is again, a dictionary. E.g. attribute "BestNights" refers to a dictionary, with each day in a week as a key and "True", "False" as value. Some attributes and categories have a null value, or the field "attribute"/"category" itself is null.

Furthermore, some fields in business.json contain a leading letter "u", e.g. u"True", which means a unicode string. We should remove letter "u" since "True" and u"True" have the same meaning.

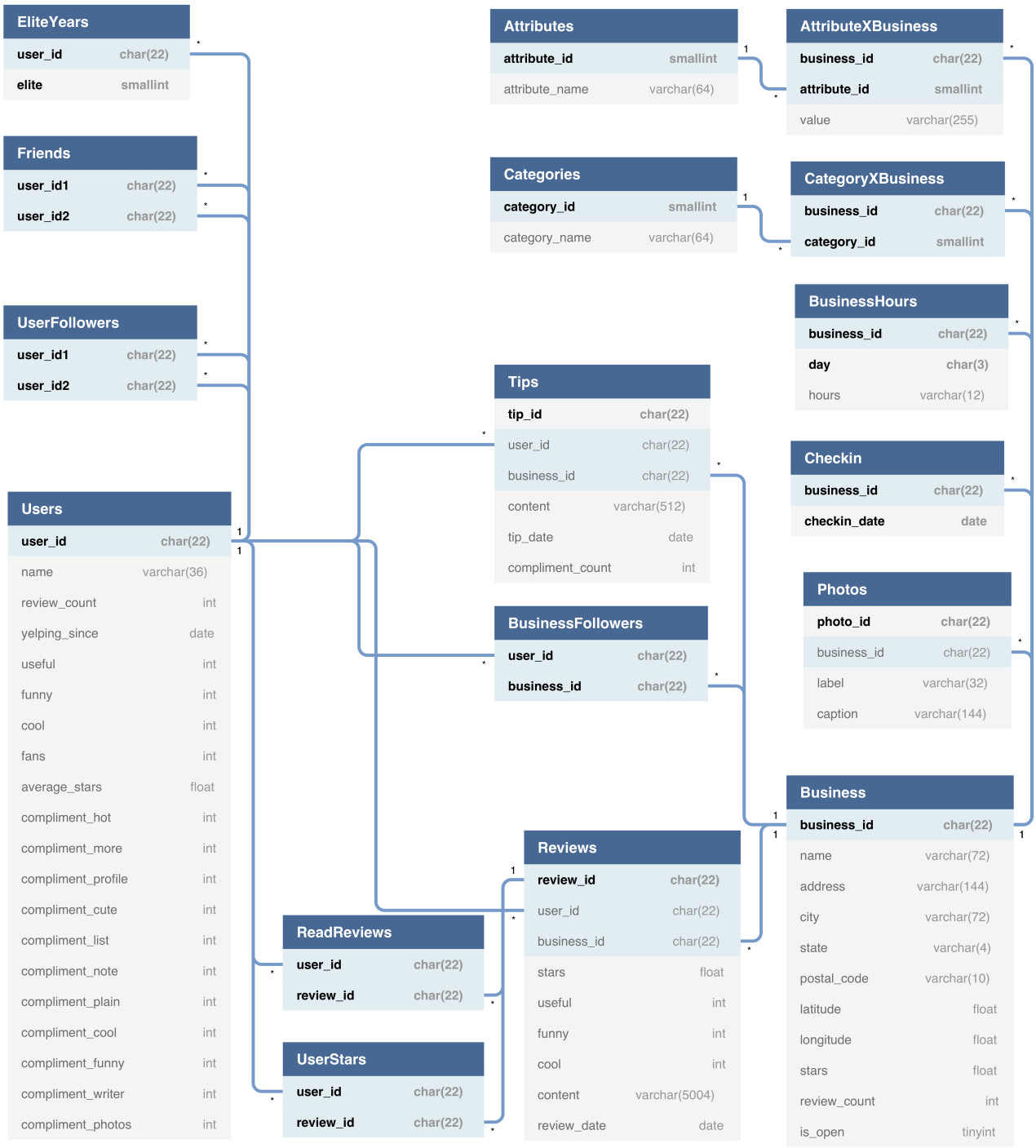
2.2 Data Cleaning

Remove all user_ids that appear only in friends list but not in "user_id" column from user.json. And remove all character "u" before texts from business.json.

3 Relational Database

3.1 Design

The ER Diagram of the database is shown as follows. The primary key of a table is in bold and the foreign keys are highlighted in light blue.



Five main tables in the diagram are *Users*, *Business*, *Reviews*, *Photos* and *Tips*. Each row of table *EliteYears* stores a user_id and a year number. Table *Friends* stores pairs of users.

Table *Attributes* and table *Categories* store ids and names of attributes/categories and connect with *Business* through junction table *AttributeXBusiness* and *CategoryXBusiness*. All values of attributes are stored as string no matter the value is "True"/"False" or a dictionary since they are not our focus in the following analysis, and spliting all of them apart will complicate the design. Table *BusinessHours* stores the hours of a business, each day per row, and days are expressed in three characters, from "Mon" to "Sun". Table *UserFollowers, BusinessFollowers, ReadReviews* and *UserStars* are used for the api and they are not part of the original data.

3.2 Dumping Data

Extract all data for tables from json files into individual files, at the same time adding required missing fields such as tip_id, and then write those files to database. Files should be read as raw string since there are lots of special characters in them.

4 Server and Client

Flask is used as the web framework.

4.1 API

14 operations are supported in the API. The status in the responsed json will be 0 if the operation succeeded, otherwise 1, with the error message in message field.

4.1.1 Login

Description:

Login as an existing user.

Method:

POST

API call:

http://127.0.0.1:5000/yelp/login

Parameters:

user_id: your user_id

API Response:


```
{
  "message": "___DPmKJsBF2X6ZKgAeGqg",
  "status": 0
}
```


4.1.2 Register

Description:

Register a new user account.

Method:

POST

API call:

http://127.0.0.1:5000/yelp/newuser

Parameters:

username: username for the new account

API Response:


```
{
 "message": "qkVPN3ZeXKxD-2y_bewUEk",
 "status": 0
}
```


4.1.3 Star/Unstar

Description:

Star/Unstar a review.

Method:

POST

API call:

http://127.0.0.1:5000/yelp/star

Parameters:

user_id: your user_id

review_id: an existing review_id

API Response:


```
{
 "message": "Starred",
 "status": 0
}
```

4.1.4 New Review

Description:

Post a new review

Method:

POST

API call:

http://127.0.0.1:5000/yelp/newpost

Parameters:

user_id: your user_id

business_id: the business_id of the business you want to comment on

content: review content

API Response:


```
{
  "message": "NnoOqAoBXsoqOWJVsTakBK",
  "status": 0
}
```


4.1.5 Follow/Unfollow a User

Description:

Follow/Unfollow a user.

Method:

POST

API call:

http://127.0.0.1:5000/yelp/followu

Parameters:

user_id1: your user_id

user_id2: user_id of the person you want to follow

API Response:


```
{
 "message": "Unfollowed",
 "status": 0
}
```


4.1.6 Follow/Unfollow a Business

Description:

Follow/Unfollow a business.

Method:

POST

API call:

http://127.0.0.1:5000/yelp/followb

Parameters:

user_id: your user_id

business_id: business_id of the business you want to follow

API Response:


```
{
 "message": "Followed",
 "status": 0
}
```

4.1.7 Get All New Reviews1

Description:

Get all new reviews by people you have followed.

Method:

GET

API call:

http://127.0.0.1:5000/yelp/uposts?u={user_id}&l={number}

Parameters:

user_id: your user_id

number(optional): number of posts to get, or empty to retrieve all

API Response:


```
{
 "message": [
  {
   "business_name": "Brake Masters",
   "review_id": "c4B3NGDZbQnRgbe4fw6qGw",
   "content": "review content 1",
   "reviewer": "Charlotte",
   "review_date": "Thu, 22 May 2014 20:20:14 GMT"
  },
  {
   "business_name": "The Watershed",
   "review_id": "GGg4KdkyvbdHwdwsC46l7g",
   "content": "review content 2",
   "reviewer": "Charlotte",
   "review_date": "Mon, 03 Aug 2015 22:26:54 GMT"
  }
 ],
 "status": 0
}
```

4.1.8 Get All New Reviews2

Description:

Get all new reviews of the business you have followed.

Method:

GET

API call:

http://127.0.0.1:5000/yelp/bposts?u={user_id}&l={number}

Parameters:

user_id: your user_id

number(optional): number of posts to get, or empty to retrieve all

API Response:


```
{
 "message": [
  {
   "business_name": "Brake Masters",
   "review_id": "c4B3NGDZbQnRgbe4fw6qGw",
   "content": "review content 1",
   "reviewer": "Charlotte",
   "review_date": "Thu, 22 May 2014 20:20:14 GMT"
  },
  {
   "business_name": "The Watershed",
   "review_id": "GGg4KdkyvbdHwdwsC46l7g",
   "content": "review content 2",
   "reviewer": "Charlotte",
   "review_date": "Mon, 03 Aug 2015 22:26:54 GMT"
  }
 ],
 "status": 0
}
```

4.1.9 My Reviews

Description:

Get all your posted reviews

Method:

GET

API call:

http://127.0.0.1:5000/yelp/mposts?u={user_id}

Parameters:

user_id: your user_id

API Response:


```
{
 "message": [
  {
   "business_name": "The Range 702",
   "content": "review content",
   "cool": 0,
   "funny": 0,
   "review_date": "Wed, 05 Aug 2015 07:20:13 GMT",
   "review_id": "Ph4MAecNE-sTQK1Fnf6E7Q",
   "stars": 5.0,
   "useful": 0
  }
 ],
 "status": 0
}
```

4.1.10 Followed Users

Description:

Get all your followed users.

Method:

GET

API call:

http://127.0.0.1:5000/yelp/followulist?u={user_id}

Parameters:

user_id: your user_id

API Response:


```
{
 "message": [
  {
   "name": "Charlotte",
   "user_id": "___DPmKJsBF2X6ZKgAeGqg"
  }
 ],
 "status": 0
}
```


4.1.11 Followed Business

Description:

Get all your followed business.

Method:

GET

API call:

http://127.0.0.1:5000/yelp/followblist?u={user_id}

Parameters:

user_id: your user_id

API Response:


```
{
 "message": [
  {
   "business_id": "__1uG7MLxWGFIv2fCGPiQQ",
   "name": "SpinalWorks Chiropractic",
   "stars": 5.0
  },
  {
   "business_id": "9Q1ZtzTPFWG4fJiFSko5Xg",
   "name": "Cantina Laredo",
   "stars": 3.5
  }
 ],
 "status": 0
}
```


4.1.12 Delete Review

Description:

Remove your posted review.

Method:

POST

API call:

http://127.0.0.1:5000/yelp/dpost?u={user_id}&r={review_id}

Parameters:

user_id: your user_id

review_id: the id of the review you want to delete

API Response:


```
{
 "message": "Success",
 "status": 0
}
```


4.1.13 Change username

Description:

Change your account username.

Method:

POST

API call:

http://127.0.0.1:5000/yelp/cn

Parameters:

user_id: your user_id

username: the username you want to set

API Response:


```
{
 "message": "Success",
 "status": 0
}
```

4.1.14 Who Am I

Description:

Display your account information.

Method:

GET

API call:

http://127.0.0.1:5000/yelp/whoami?u={user_id}

Parameters:

user_id: your user_id

API Response:


```
{
 "message": [
  {
   "cool": 0,
   "fans": 0,
   "funny": 0,
   "name": "Mike",
   "reviews": 5,
   "stars": 5.0,
   "useful": 1,
   "user_id": "___QCazm0YrHLd3uNUPYMA",
   "yelping_since": "Thu, 31 Jul 2014 21:53:24 GMT"
  }
 ],
 "status": 0
}
```

4.1.15 Reset

Description:

Reset read history so that you can see the read reviews again.

Method:

POST

API call:

http://127.0.0.1:5000/yelp/reset

Parameters:

user_id: your user_id

API Response:


```
{
 "message": "Success",
 "status": 0
}
```

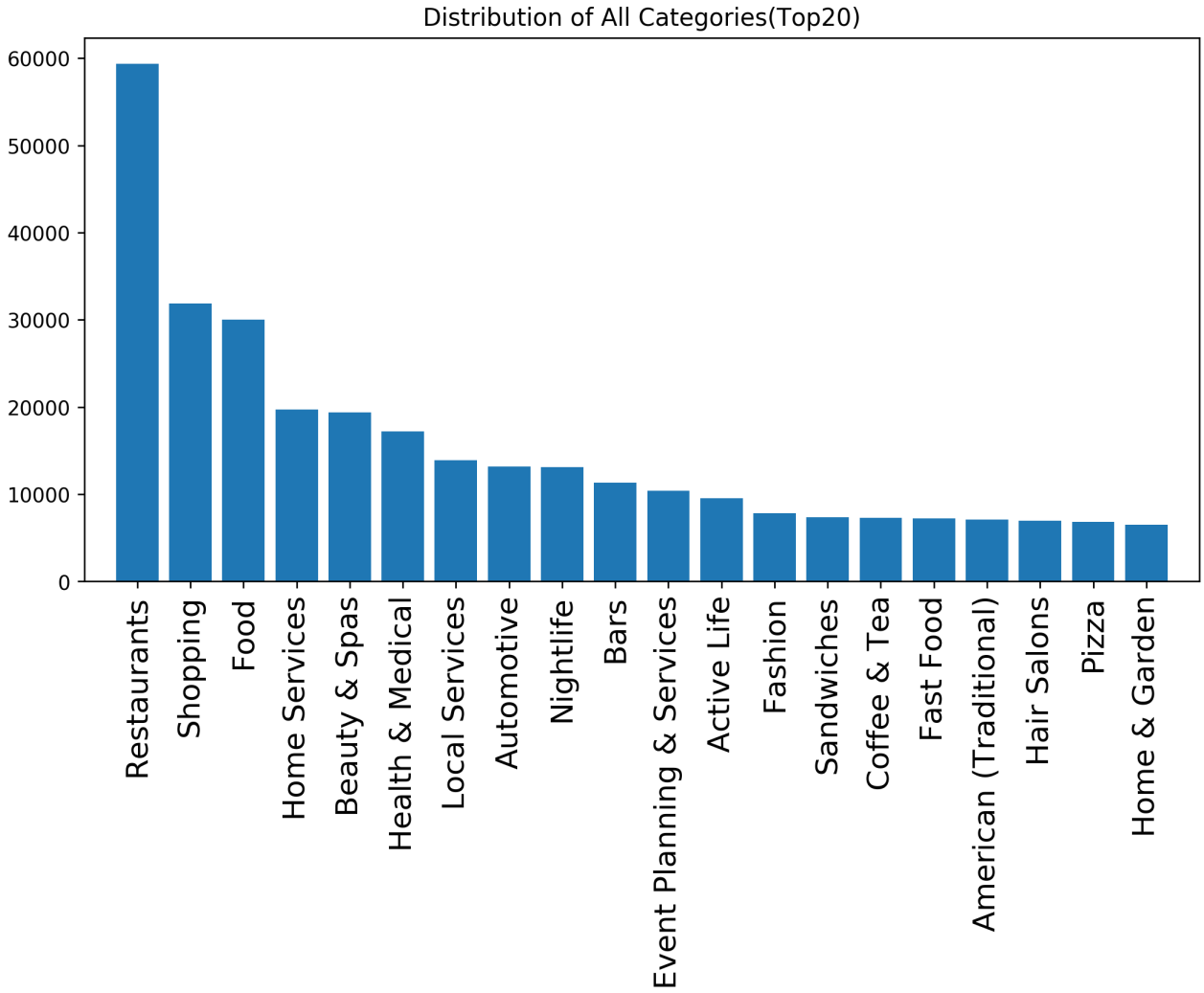

4.2 Client

A simple client to test the api.

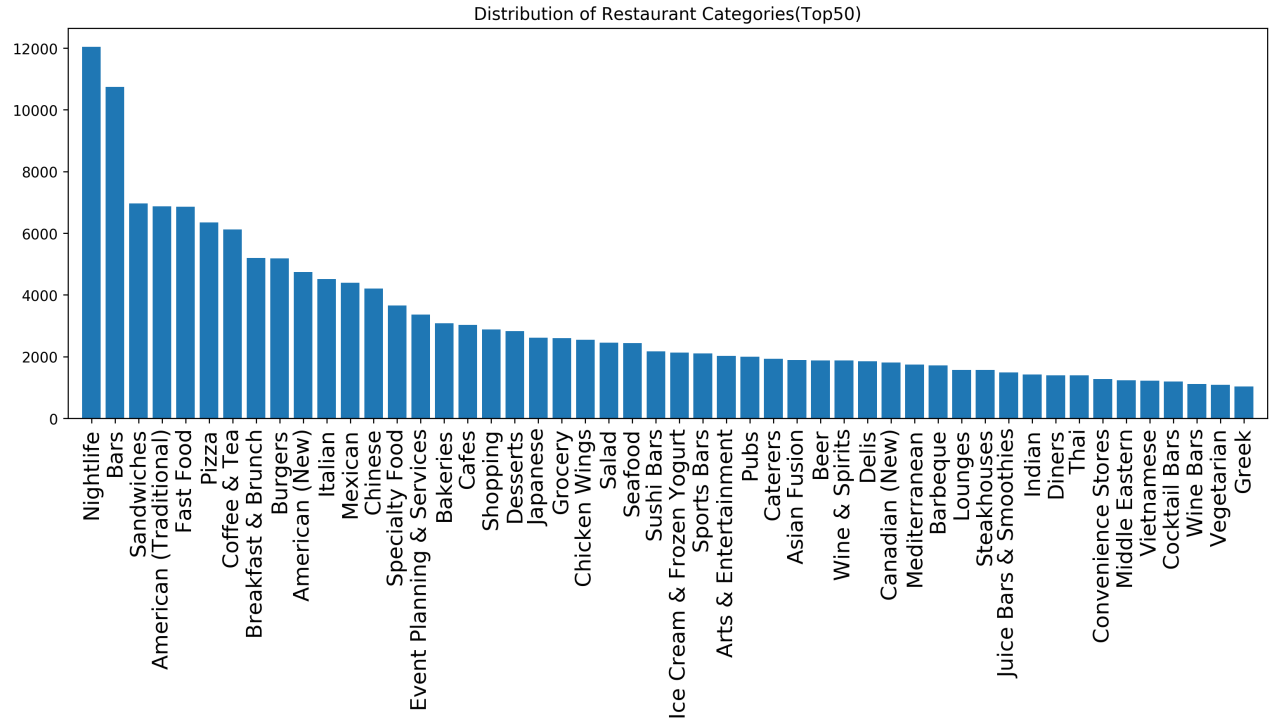
5 Data Mining

5.1 Data Visualization

5.1.1 Top 20 Categories in Business

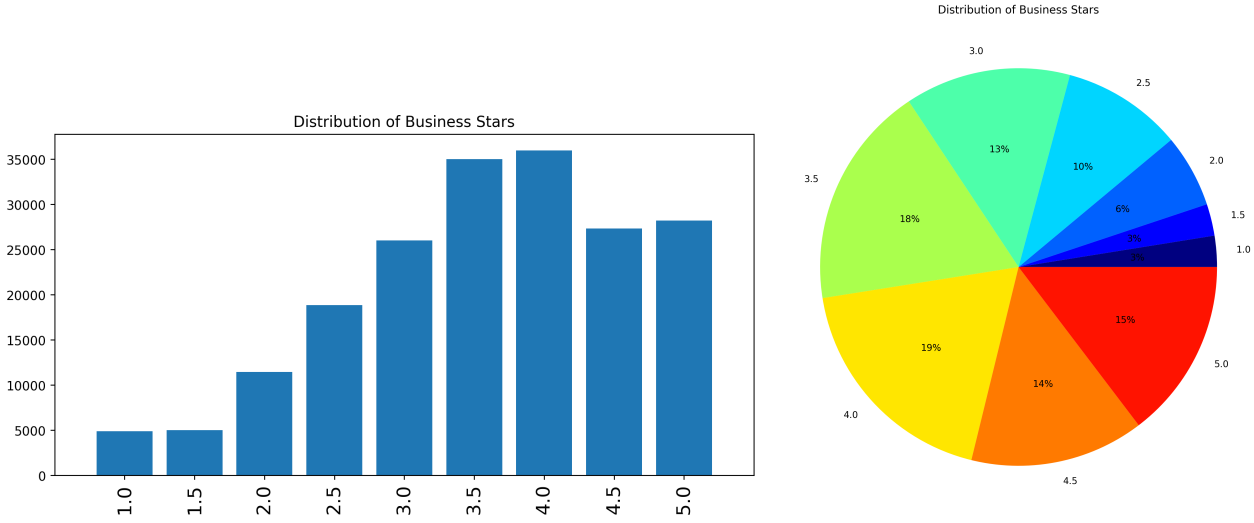


5.1.2 Top 50 Categories in Restaurants



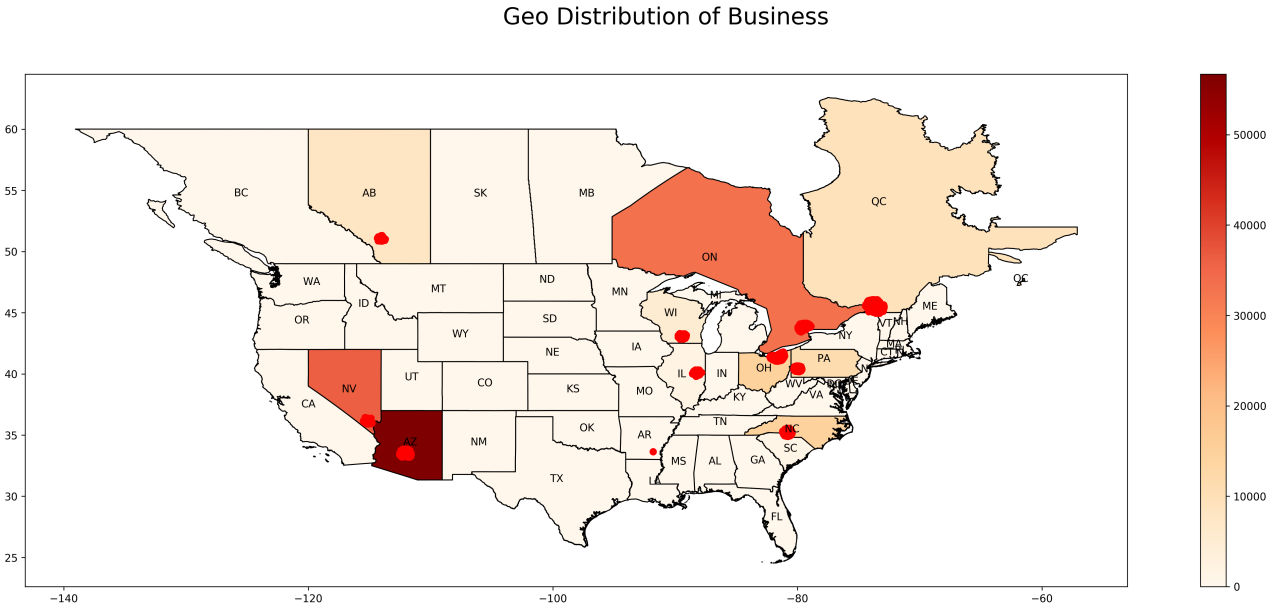
Since basically every restaurant has attribute *Restaurant* and *Food*, we plot the graph starting from the third most attribute.

5.1.3 Distribution of Business Score



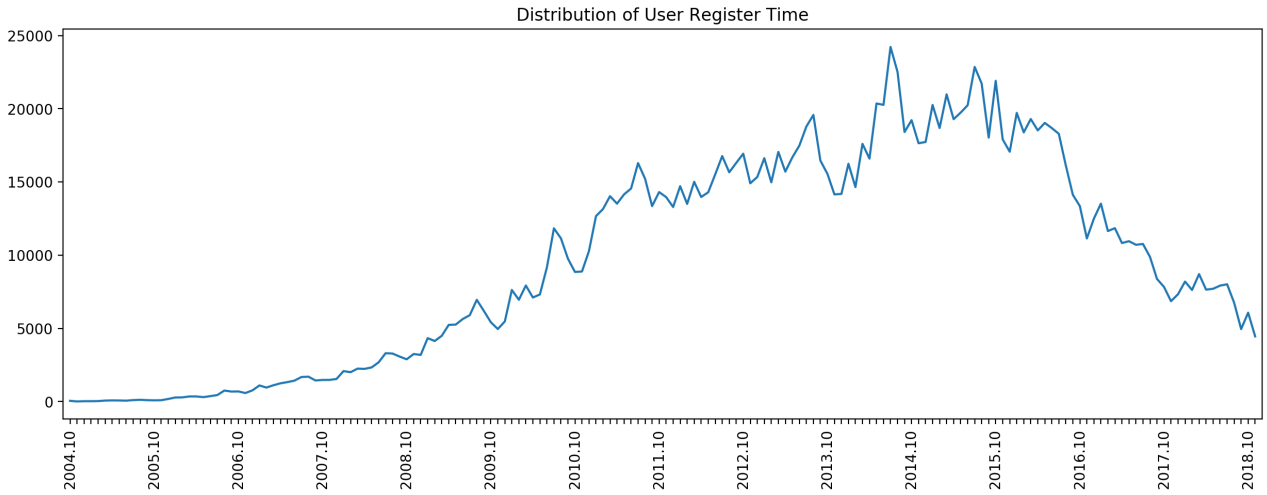
About 80 percent of the business has a score greater or equal than 3.0.

5.1.4 Geographical Distribution of the Business



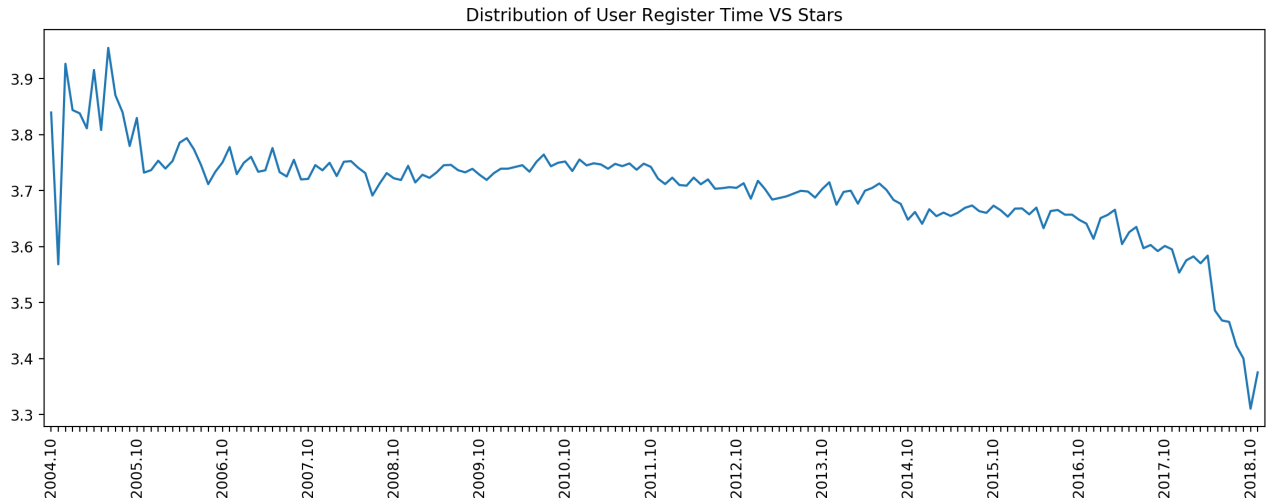
We can get from the plot where are the 10 metropolitan areas that the data came from. Most of them are from the USA and Canada, a small amount of reviews from Alaska and the UK are not shown on this plot.

5.1.5 Distribution of Registered Users Over Time



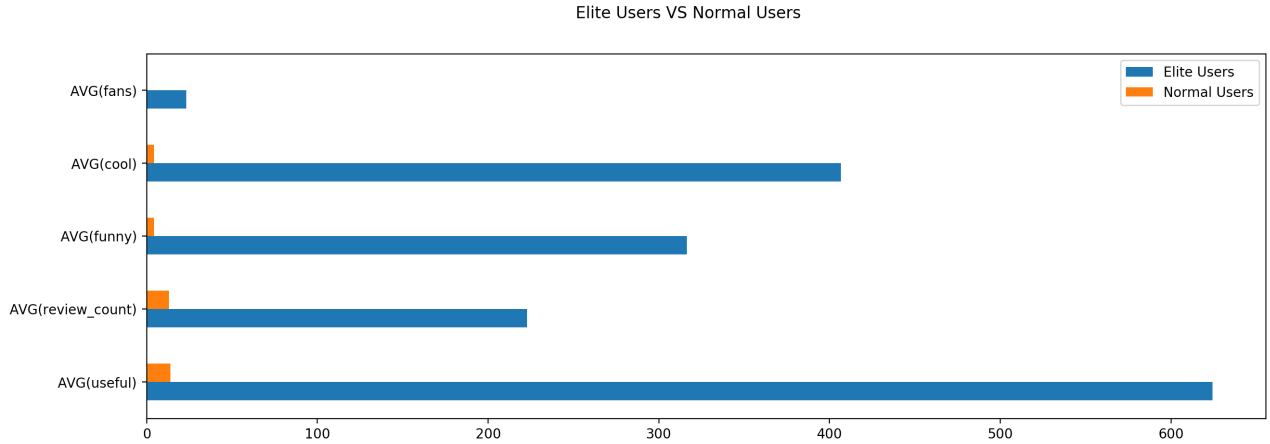
The peak of the user register time is from 2014 to 2016.

5.1.6 User Satisfaction VS User Register Time



The newer a user is, the more likely he/she will give a lower score to a business, which means the users are more and more difficult to satisfy over time.

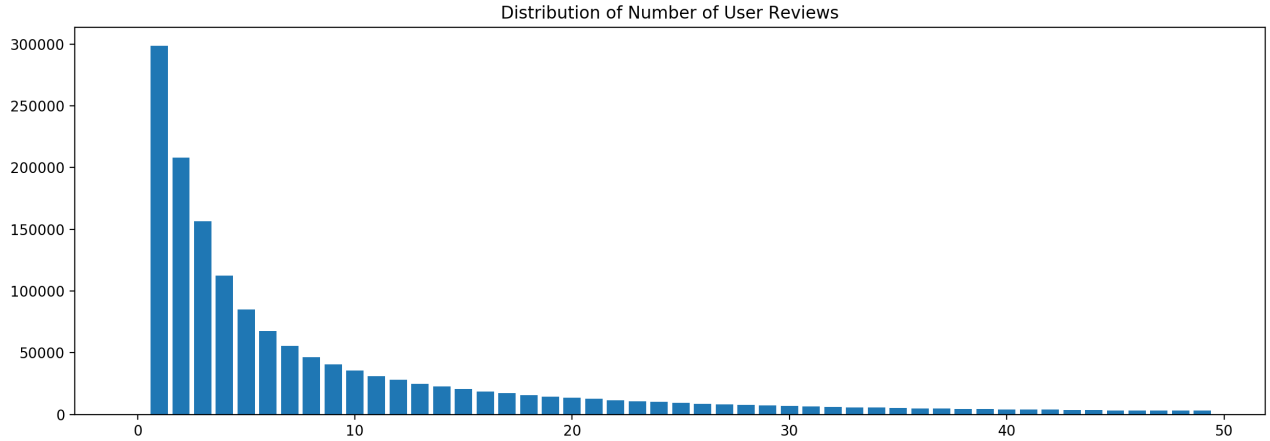
5.1.7 Comparison Between Normal Users and Elite Users



On average, an elite user has more than 600 useful count, 200 review count, 300 funny count, 400 cool count and 20 fans, which are dozens to hundreds of times greater than those of the

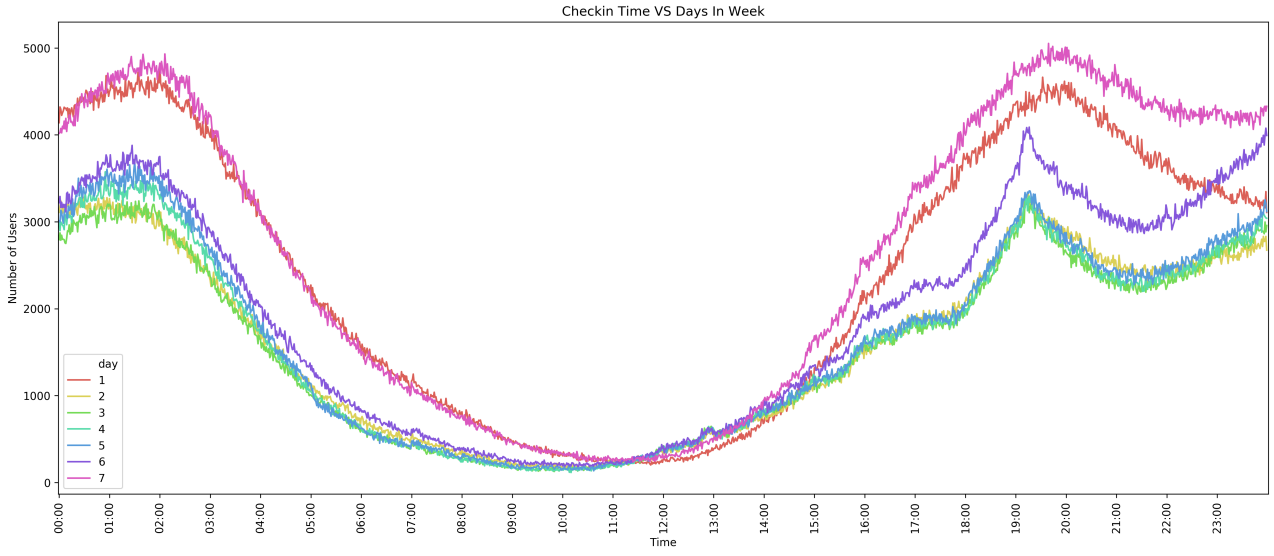
normal users. It is indeed not easy to become an elite user.

5.1.8 Distribution of the User Review Number



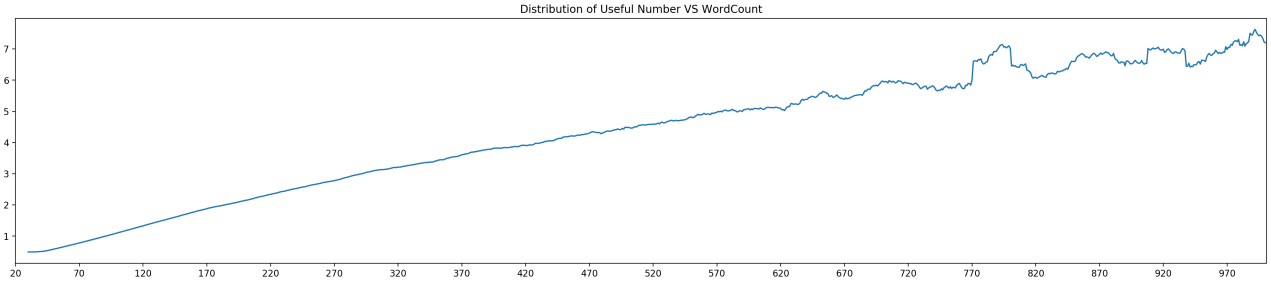
More than 90 percent of users have less or equal than 10 reviews.

5.1.9 User Checkin Time for Each Day in a Week



Users are more likely to check in on a business from 7PM to 2 AM, which is reasonable because people were off work and finished their dinner around that time. And there are most checkins on Sunday and Monday, which is probably because people went to that restaurant or store on Saturday and Sunday, and one day later they wrote reviews about that business.

5.1.10 Useful number VS Word Count in a Review



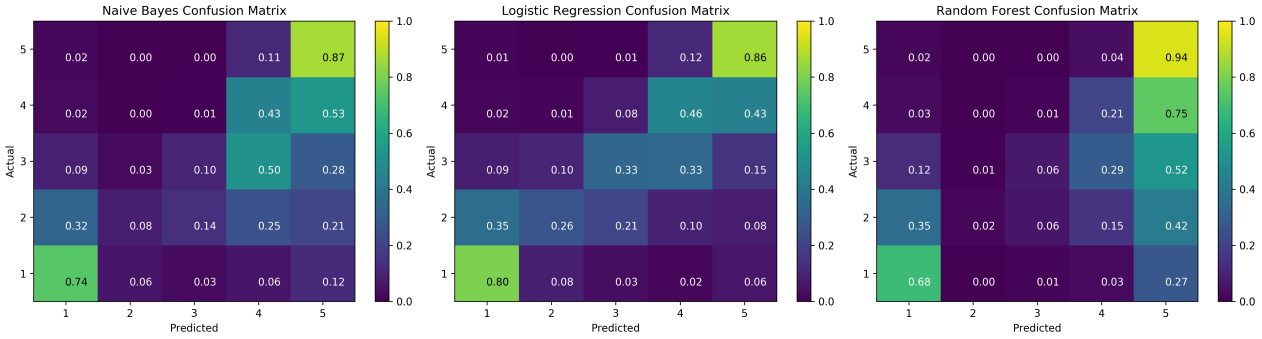
The more words in a review, the more likely it is tagged as useful.

5.2 Classification

We perform classification on review stars(1.0 to 5.0) based on the content of a review. 125,000 reviews are selected for the classification, in which 80% are used for training and 20% are used for testing. We cannot preform classifiers on texts directly, so the texts are converted to a matrix of TF-IDF features using TfidfVectorizer in sklearn.

5.2.1 Classification without PCA

Apply Naive Bayes Classifier, Logistic Regression and Random Forest Classifier on the training data. (Timeout on other classifiers.) Then plot their confusion matrices using testing data.



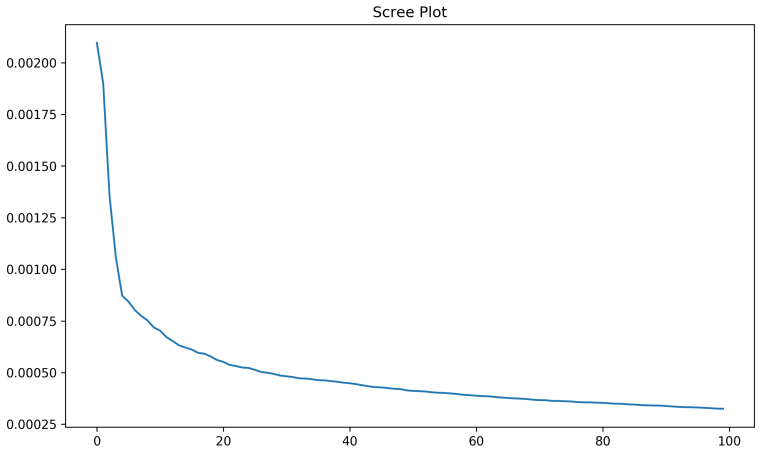
We can see that all of these three classifiers can classify the 1 and 5 stars reviews really well. But they tend to predict 4 and 3 stars and sometimes even 2 stars reviews as 5 stars reviews. And they also tend to predict 2 stars reviews as 1 star reviews. Overall, 2 stars reviews are the most difficult ones to predict, following by 3 stars and 4 stars because the words used in those reviews are really similar to each other and sometimes even a human cannot tell which class they should be in. If we specify that a review that greater or equal to 3 stars is good and otherwise is bad, and do a binary classification, the result will be much better.

We can get the most positively related words and the most negatively related words in the review from the trained logistic regression model. Those words play a decisive role in the classification. Their wordclouds with top 20 words are shown below:

For random forest classifier, we cannot distinguish the "best" and the "worst" words, but we can still get the crucial words. The wordcloud with top 30 words are shown below:

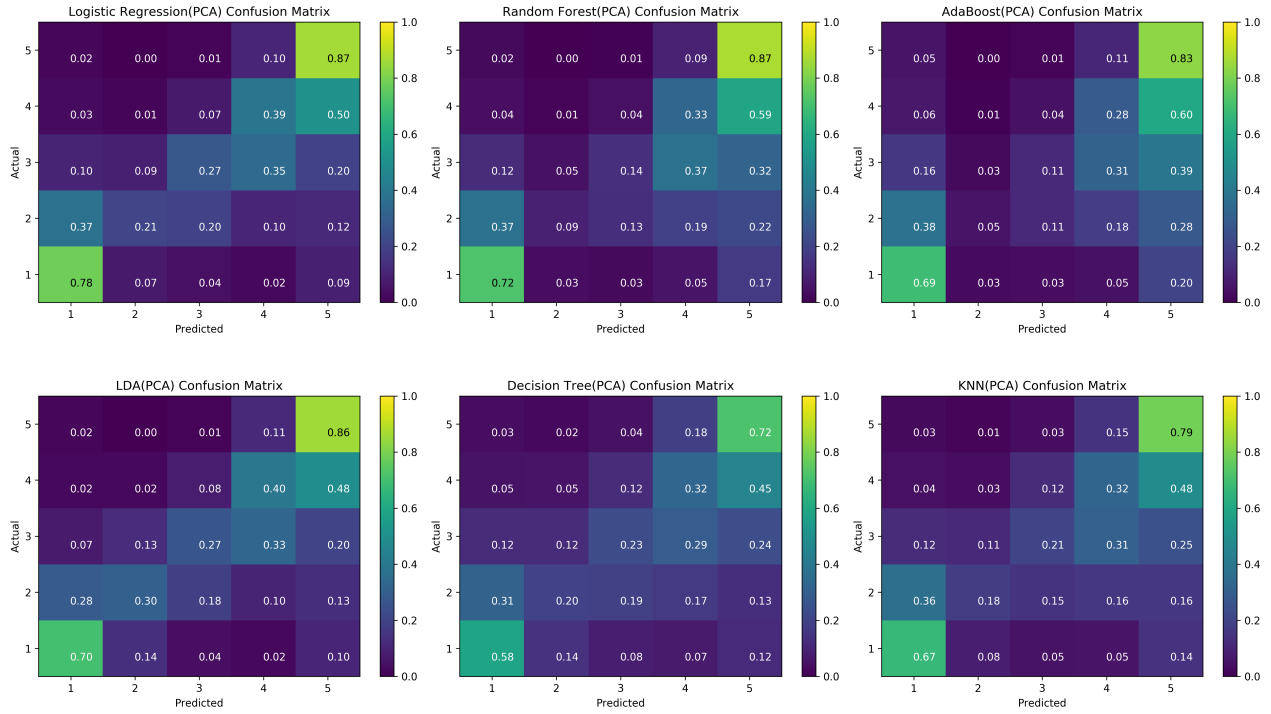
5.2.2 Classification with PCA

The dimension of the raw data is rather high, making it impossible to run some algorithms. We first run standard scalar on the data, which is required by PCA. Then we draw the scree plot of PCA.



From the scree plot we know that each of the selected components explains a really small percentage of varince. Based on the plot, we use n_components=50 to process our data.

Then, we apply logistic regression, random forest, adaboost, LDA, decision tree and KNN classifiers on it. (PCA makes some value negative, therefore we cannot use Naive Bayes classifier. Also, Naive Bayes based on applying Bayes' theorem with strong independence assumptions between the features, and preprocessing will seriously impact the result.) Their confusion matrices are as below:



Desision Tree classifier generates the worst result. It is merely better than random guess when predicting the 2, 3 and 4 stars revivews.

The performance of KNN and Adaboost is average. After PCA, the accuracy of both Logistic Regreesion and Random Forest decreases a bit. But Logistic regression is still the best method to split different reviews. And LDA is also a good method on this problem.

However, most classifiers run under default parameters, with grid search to find better pa-

rameters and other preprocessing methods, the performance of these classifiers may be better.