ECE656 Project – Yelp Dataset

Tiankai Jiang

March 11, 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 user-name 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.
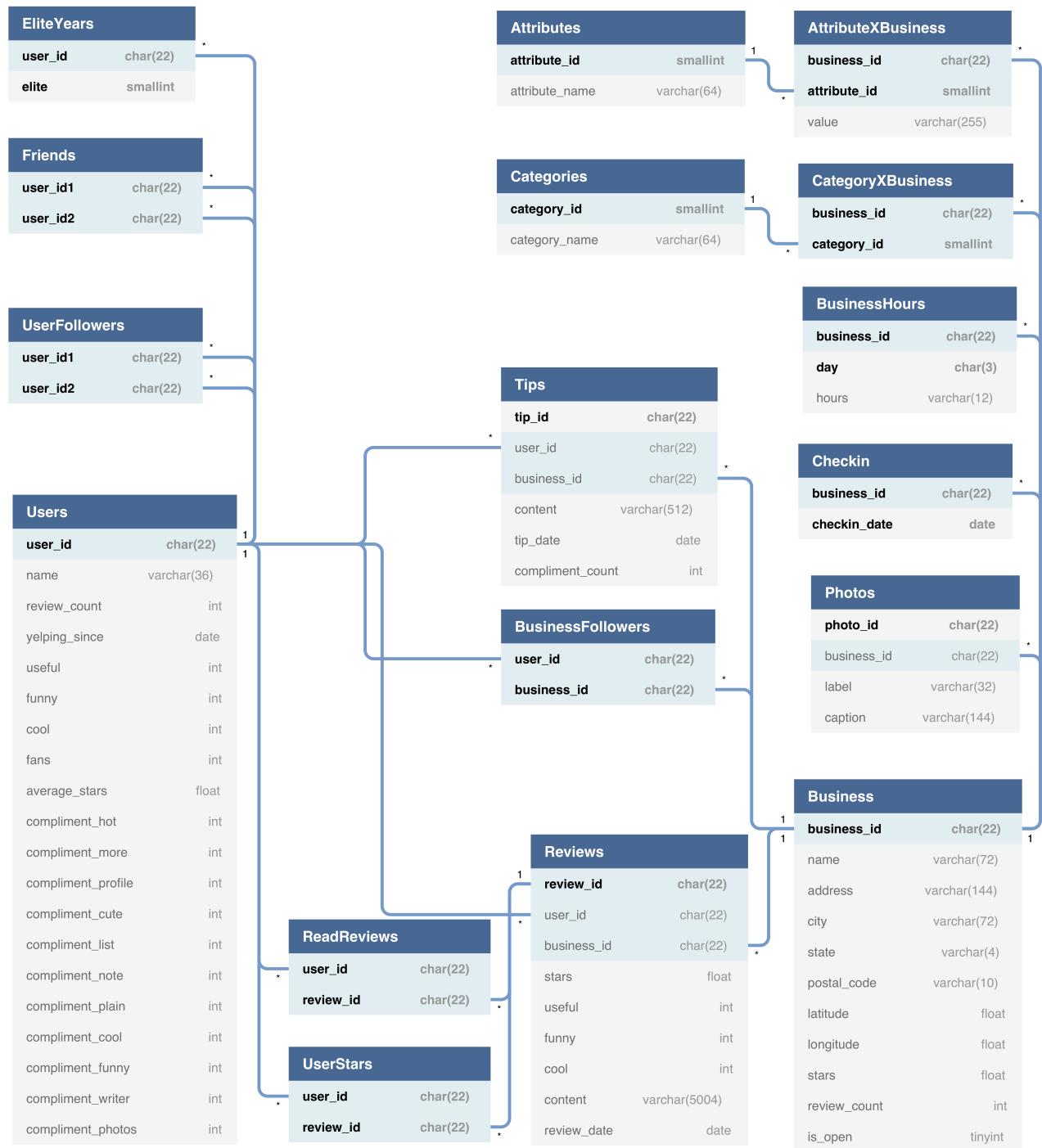
2.2 Data Cleaning

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.

EliteYears

user_id	char(22)
elite	smallint

Friends

user_id1	char(22)
user_id2	char(22)

UserFollowers

user_id1	char(22)
user_id2	char(22)

Users

user_id	char(22)
name	varchar(36)
review_count	int
yelping_since	date
useful	int
funny	int
cool	int
fans	int
average_stars	float
compliment_hot	int
compliment_more	int
compliment_profile	int
compliment_cute	int
compliment_list	int
compliment_note	int
compliment_plain	int
compliment_cool	int
compliment_funny	int
compliment_writer	int
compliment_photos	int

ReadReviews

user_id	char(22)
review_id	char(22)

UserStars

user_id	char(22)
review_id	char(22)

Attributes

attribute_id	smallint
attribute_name	varchar(64)

Categories

category_id	smallint
category_name	varchar(64)

Tips

tip_id	char(22)
user_id	char(22)
business_id	char(22)
content	varchar(512)
tip_date	date
compliment_count	int

BusinessFollowers

user_id	char(22)
business_id	char(22)

Reviews

review_id	char(22)
user_id	char(22)
business_id	char(22)
stars	float
useful	int
funny	int
cool	int
content	varchar(5004)
review_date	date

AttributeXBusiness

business_id	char(22)
attribute_id	smallint
value	varchar(255)

CategoryXBusiness

business_id	char(22)
category_id	smallint

BusinessHours

business_id	char(22)
day	char(3)
hours	varchar(12)

Checkin

business_id	char(22)
checkin_date	date

Photos

photo_id	char(22)
business_id	char(22)
label	varchar(32)
caption	varchar(144)

Business

business_id	char(22)
name	varchar(72)
address	varchar(144)
city	varchar(72)
state	varchar(4)
postal_code	varchar(10)
latitude	float
longitude	float
stars	float
review_count	int
is_open	tinyint

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

8 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.

API call:

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

Parameters:

user_id: your user_id

API Response:

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

4.1.2 Register

Description:

Register a new user account.

API call:

http://127.0.0.1:5000/yelp/newuser?n={username}

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.

API call:

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

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

API call:

http://127.0.0.1:5000/yelp/newpost?u={user_id}&b={business_id}&r={content}

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.

API call:

http://127.0.0.1:5000/yelp/followu?u={user_id1}&f={user_id2}

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.

API call:

http://127.0.0.1:5000/yelp/followb?u={user_id}&b={business_id}

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.

API call:

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

Parameters:

user_id: your user_id

API Response:

```
{
 "message": [
  {
   "business_id": "Review content1",
   "reviewer": "___DPmKJsBF2X6ZKgAeGqg"
  },
  {
```

```
    "business_id": "Review  content2",
    "review_id": "GGg4KdkyvbdHwdwsC46l7g",
    "reviewer": "___DPmKJsBF2X6ZKgAeGqg"
  }
],
"status": 0
}
```

4.1.8   Get All New Reviews2

Description:

Get all new reviews of the business you have followed.

API call:

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

Parameters:

user_id: your user_id

API Response:

```
{
  "message": [
    {
      "business_id": "Review  content1",
      "reviewer": "___DPmKJsBF2X6ZKgAeGqg"
    },
    {
      "business_id": "Review  content2",
      "review_id": "GGg4KdkyvbdHwdwsC46l7g",
      "reviewer": "___QCazm0YrHLd3uNUPYMA"
    }
  ],
  "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

Distribution of All Categories(Top20)
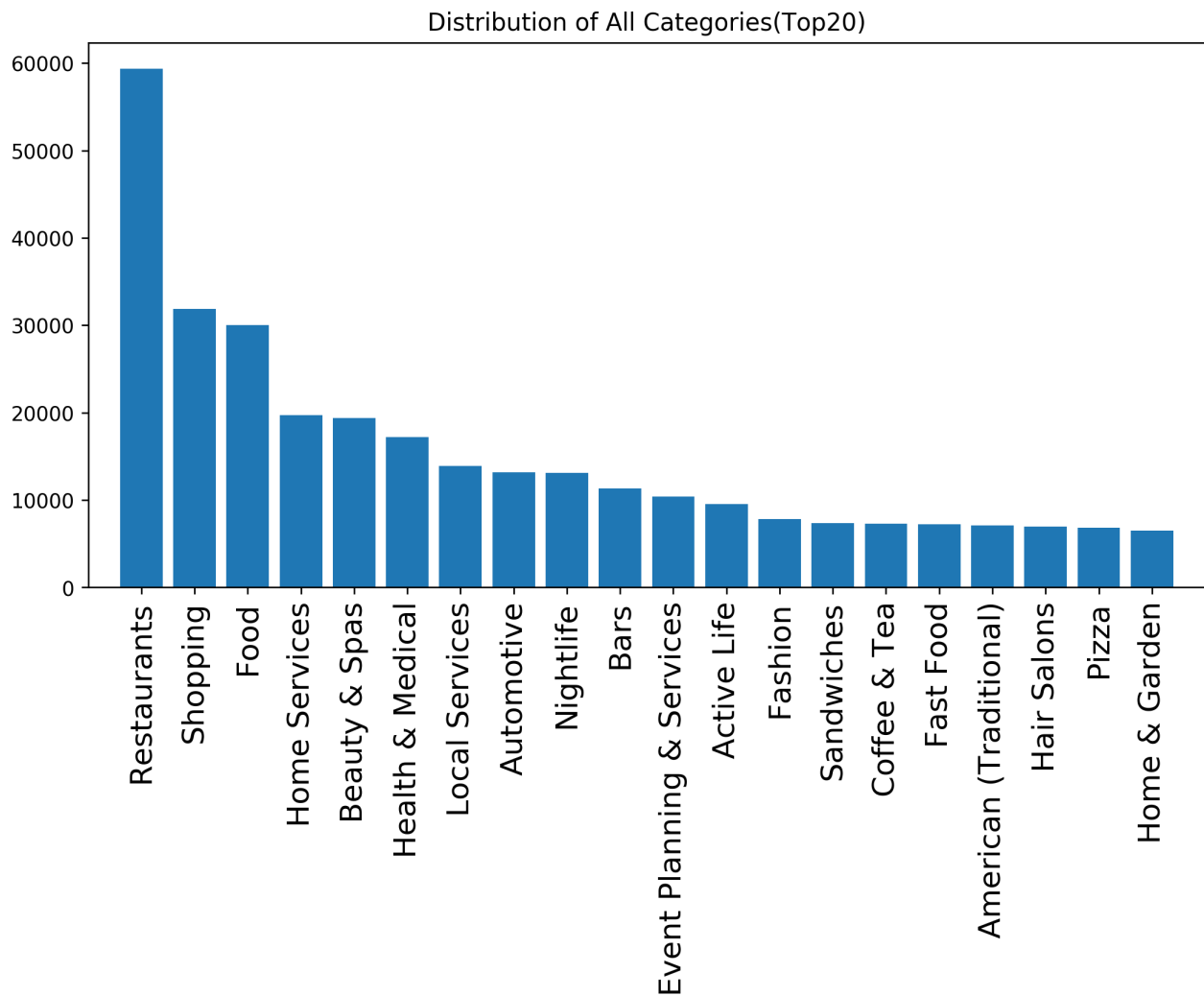
5.1.2 Top 50 Categories in Restaurants

Distribution of Restaurant Categories(Top50)
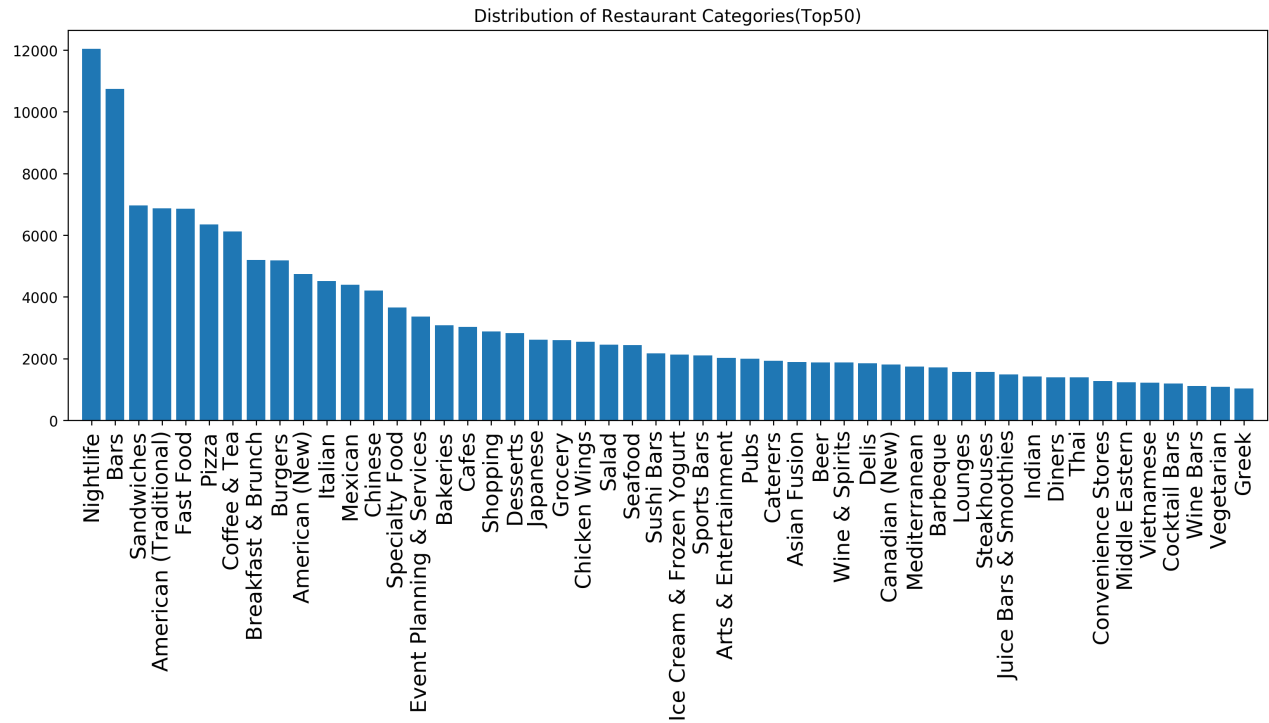
Since basically every restaurant has attribute *Restaurant* and *Food*, we plot the graph starting

from the third most attribute.

5.1.3 Distribution of Business Score
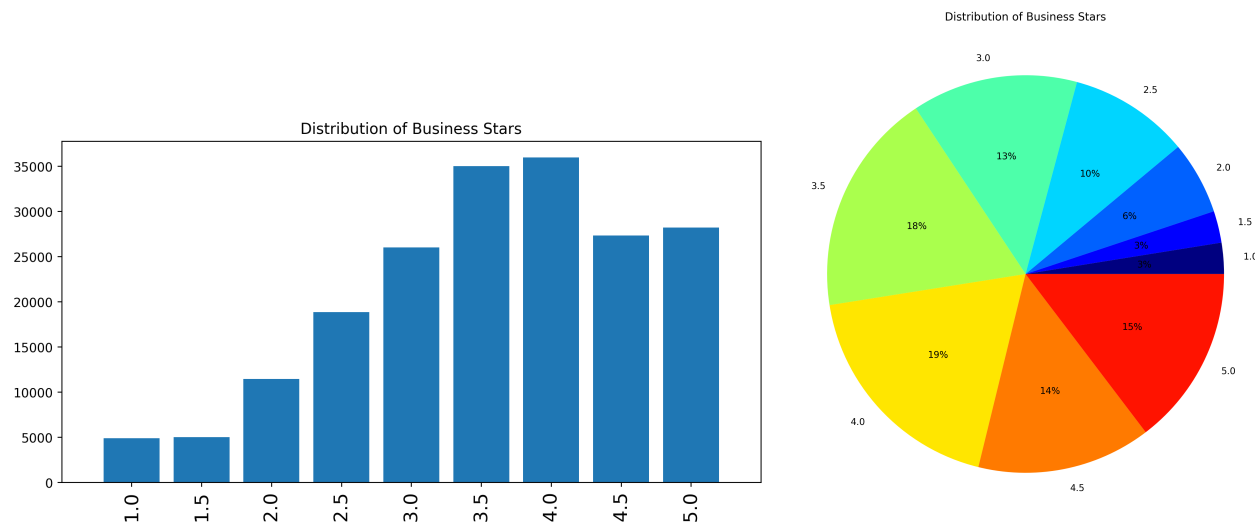
About 80 percent of the business has a score greater or equal than 3.0.

5.1.4 Geographical Distribution of the Business

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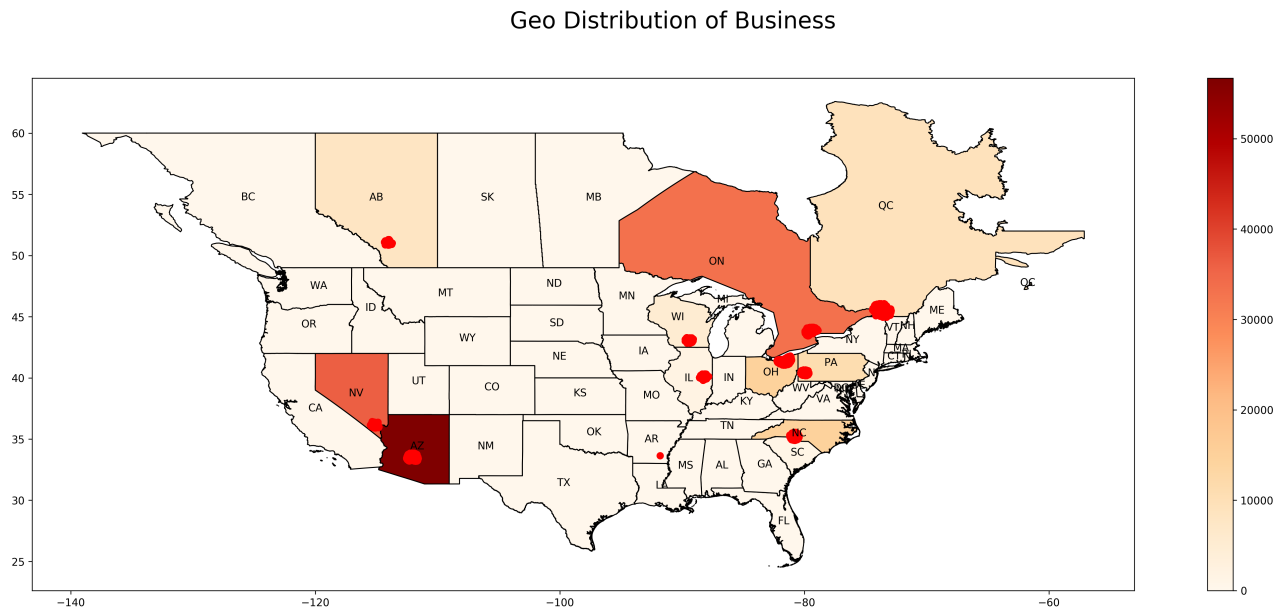
We can get from the plot where are the 10 metropolitan areas that the data came from. Most of them are from the USA and Canada, a small amount of reviews from Alaska and the UK are not shown on this plot.

5.1.5   Distribution of Registered Users Over Time
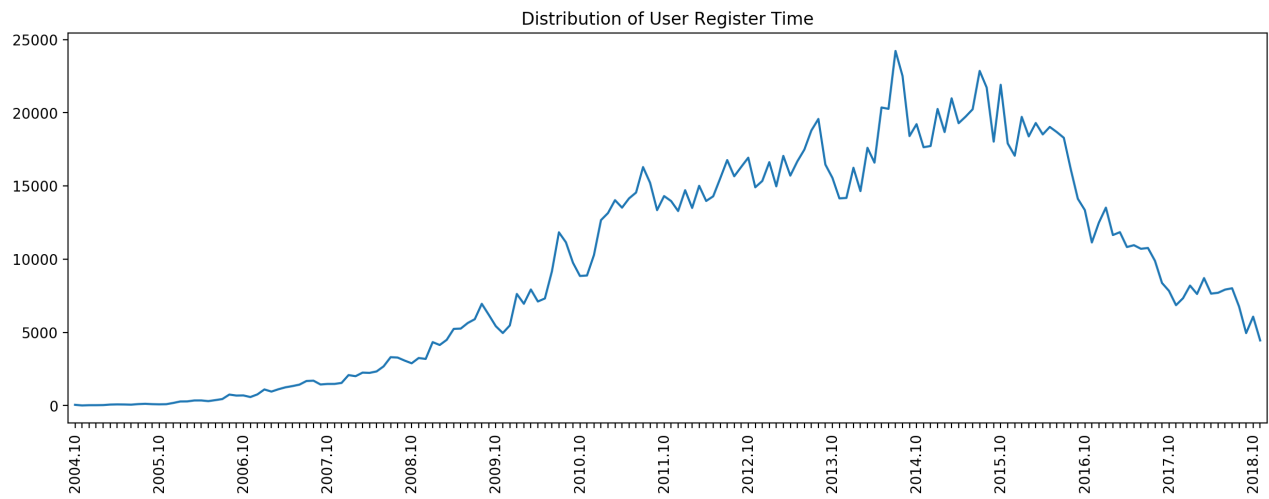
The peak of the user register time is from 2014 to 2016.

5.1.6 User Satisfaction VS User Register Time

Distribution of User Register Time VS Stars
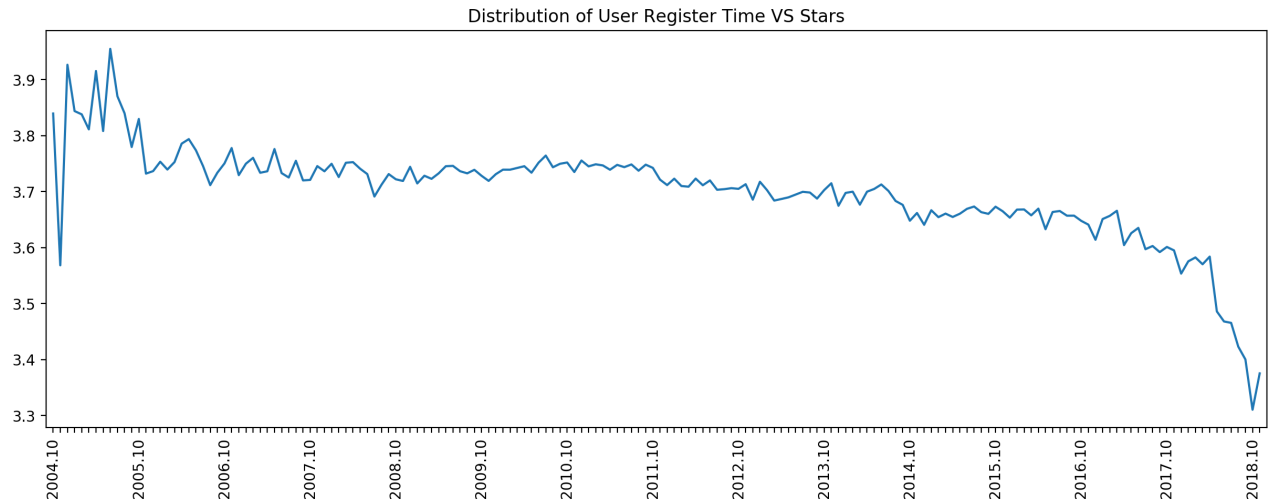
The newer a user is, the more likely he/she will give a lower score to a business, which means the users are more and more difficult to satisfy over time.
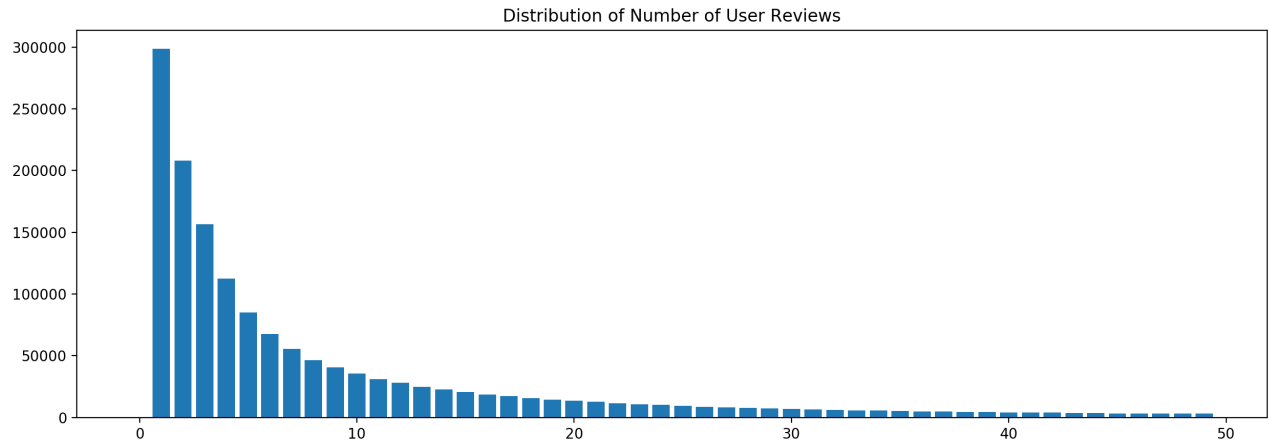
5.1.7 Comparison Between Normal Users and Elite Users

Elite Users VS Normal Users
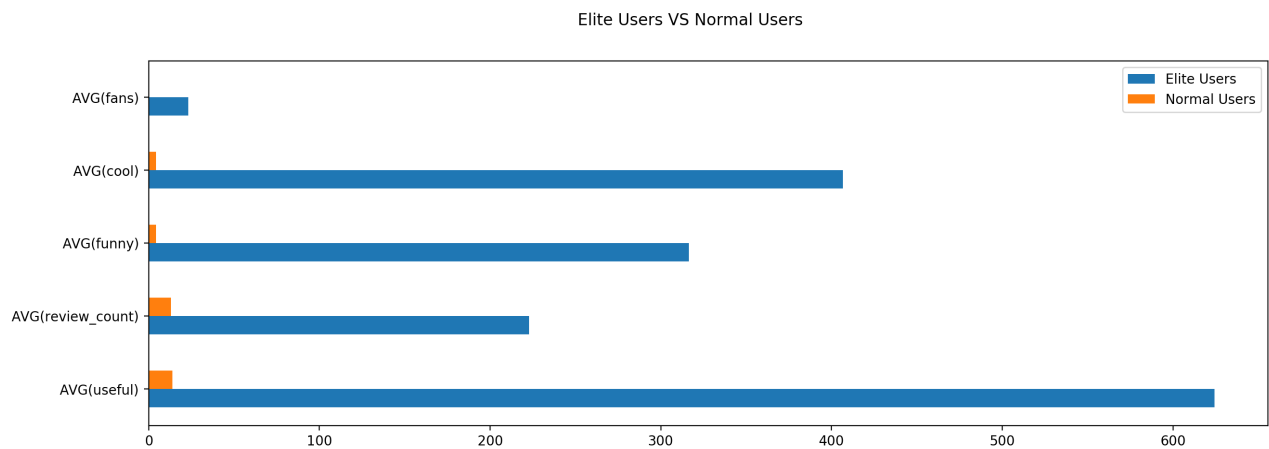
On average, an elite user has more than 600 useful count, 200 review count, 300 funny count, 400 cool count and 20 fans, which are dozens to hundreds of times greater than those of the
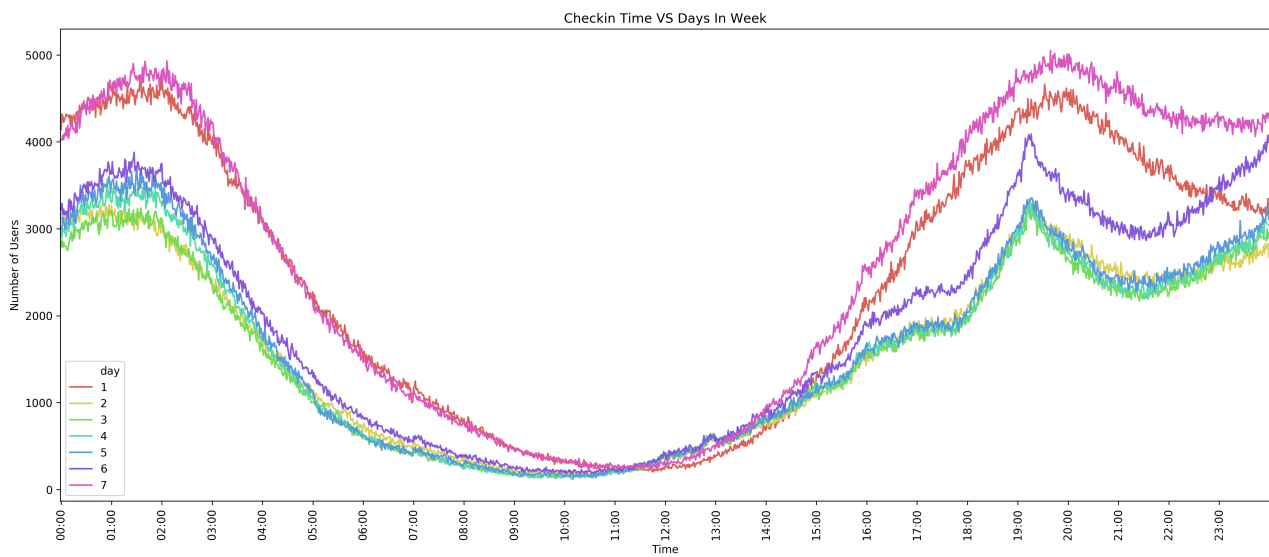
14

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

Distribution of Useful Number VS WordCount
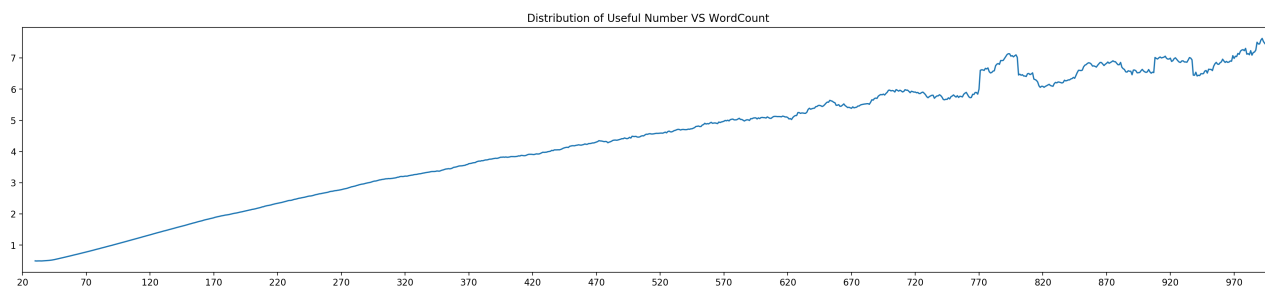
The more words in a review, the more likely it is tagged as useful.

5.2 Classification

We perform classification on review stars(1.0 to 5.0) based on the content of a review.