Where should I eat tonight?

Fredrik Emilsson

Thursday, November 19, 2015

Introduction

The goal for this Data Science Capstone was to create a question that are answerable based on the delivered data from Yelp. After that find out how it is possible to answer this question.

I decided to create a question that contains two steps. First I want to predict how many stars a user would give a specified restaurant. In a second step the usage of this prediction will answer the following question: I am in a specific location and wants to know which restaurant within a specified distance I should visit for my dinner.

In this report I will describe the way I took to answer those questions, which steps I needed to have to get the result and how good the answers to the questions are.

Methods and Data

How should you tackle to get good estimation about how many stars a user will give to a specified restaurant and then rank the restaurants in the neighborhood?

A good library I used a lot during the capstone was **sqldf** that you will see examples of during this report.

Uniform the data

To get a more uniformed data format I decided to do some manipulation of the input data. Specially regarding the business dataset. To get rid of lists and deep name-values I changed both neighborhoods, categories and attributes to be normal name-value objects in the json file. Here are two examples:

"categories": ["Nightlife", "Belgian"] => "categories": {"Nightlife": 1, "Belgian": 1}

"attributes": {"Outdoor Seating": false,"Price Range": 1} => "attributes": {"Outdoor Seating false": 1, "Price Range 1": 1}}

This also simplifies the prediction if all interesting features are numbers, 1 if exists.

This part did I do by a Scala object which uniformed the business.json-file.

Understand the data

You have to understand which data there are and how to use it to answer the questions. There are five different datasets: **business**, **review**, **user**, **check-in** and **tip**. Which of them contains interesting data? Three of them are definitely interesting to use to be able to answer the question:

- **Business** since it contains information about the restaurants. Some of them are business_id, name, full_address, latitude, longitude, stars, review_count, categories and attributes.
- **User** since it contains information about the users. Some of them are user_id and average_stars.
- **Review** since it combines the restaurants with their users. Some of them are business_id, user_id, stars and date.

I decided to make a join between business, review and user as the base data set. It was also filtered so only restaurants are included. This base data set is then derived to produce more interesting information as I will show in next chapter.

To make the prediction to be as good as possible I calculate the real average stars for a business's, instead of the one that was rounded to half-stars, by using the review dataset. This was then joined with the base data set.

The total dim of the join are 990627 rows and 603 columns.

Here is parts of how I did this in practice:

```
business <- fromJSON(paste("[", paste(readLines(business_path), collapse=","), "]"),
                     flatten = TRUE); names(business) <- updateNames(names(business), "b")
review <- fromJSON(paste("[", paste(readLines(review_path), collapse=","), "]"),
                   flatten = TRUE); names(review) <- updateNames(names(review), "r")
user <- fromJSON(paste("[", paste(readLines(user_path), collapse=","), "]"),
                 flatten = TRUE); names(user) <- updateNames(names(user), "u")

business_restaurants <- sqldf("select * from business where b_categories__Restaurants=1")
review_restaurants <- sqldf("select distinct r.* from business_restaurants b, review r
                              where r_business_id=b_business_id")
user_restaurants <- sqldf("select distinct u.* from review_restaurants r, user u
                           where r_user_id=u_user_id")
mean_stars <- sqldf("select r_business_id,avg(r_stars) from review_restaurants
                     group by r_business_id"); names(mean_stars) <- c("m_business_id","m_stars")
restaurants_join <- sqldf("select * from review_restaurants r, user_restaurants u,
                          business_restaurants b, mean_stars m
                          where r_user_id=u_user_id and r_business_id=b_business_id and
                          r_business_id=m_business_id")
```

Understand which type of restaurant a user likes

Maybe the most important derivation of the data is to calculate which of the restaurants categories and attributes a user likes or dislikes. During my report I will when I talk about **categories** and **attributes** call it **types**.

If a user gives a restaurant a higher star than the average user, he or she probably likes this kind of restaurants and therefor will other similar restaurants with same types also probably get a higher star than average. A similar process is done when a user dislikes a restaurant. By using this knowledge (which you will see below) it will improve the prediction of review stars for restaurants.

Here are the steps I used to generate how good a user likes a type and then use it against restaurants:

1. Create a data frame that contains all types, average star for the restaurant (b_stars), number of starts the user gave on the review (r_stars) and id of the user from the active data set (user_id). All NA related to a types are set to 0.

```
##  bstars rstars                 userid type1 type2             businessid
##     3.0      4 4jGnjQUArjFKOnxShpGLXw     0     0 YaOoNqP8mzPU5HwMdgf5zQ
##     4.0      5 4jGnjQUArjFKOnxShpGLXw     1     1 cmGR1HS9ms233roSllcglw
##     4.5      4 8fApIAMHn2MZJFUiCQto5Q     1     0 2X5G4Ujq0s4Wfn4TC7gXOg
##     4.0      3 8fApIAMHn2MZJFUiCQto5Q     0     1 nVyL22lcEZoUNoAMgfKOyQ
```

2. For each review get the difference between the review stars with the average stars for the restaurant (rstars - bstars) and multiply all the types with this value. Then join this with userid:

```
user_data <- (ranking$rstars-ranking$bstars)*ranking[,4:(ncol(ranking)-1)]
df <- data.frame(ranking[,3],user_data); names(df)[1] <- "userid"
print(df[c(34,13,4,32),], row.names = FALSE)
```

```
##                  userid type1 type2
##  4jGnjQUArjFKOnxShpGLXw   0.0     0
##  4jGnjQUArjFKOnxShpGLXw   1.0     1
##  8fApIAMHn2MZJFUiCQto5Q  -0.5     0
##  8fApIAMHn2MZJFUiCQto5Q   0.0    -1
```

3. Aggregate by userid to get the average value for each type for each user (in the average only values that is not 0 is used).

```
user_rank_mean <- aggregate(df[,2:ncol(df)], by=list(df$userid), meanOfNot0)
names(user_rank_mean)[1] <- "userid"
print(user_rank_mean, row.names = FALSE)
```

```
##                  userid type1  type2
##  4jGnjQUArjFKOnxShpGLXw   1.0 -0.250
##  8fApIAMHn2MZJFUiCQto5Q  -0.5 -0.625
```

4. Join the aggregate result with the base data set.

```
## Loading required package: tcltk
```

```
##                  userid type1  type2 btype1 btype2             businessid
##  4jGnjQUArjFKOnxShpGLXw   1.0 -0.250      0      0 Ya0oNqP8mzPU5HwMdgf5zQ
##  4jGnjQUArjFKOnxShpGLXw   1.0 -0.250      1      1 cmGR1HS9ms233roSllcglw
##  8fApIAMHn2MZJFUiCQto5Q  -0.5 -0.625      1      0 2X5G4Ujq0s4Wfn4TC7gX0g
##  8fApIAMHn2MZJFUiCQto5Q  -0.5 -0.625      0      1 nVyL22lcEZoUNoAMgfKOyQ
```

5. For each row and each type multiply the aggregated value with the restaurants types.

```
tp1 <- restaurants_join_rank[,c(2,3)]; tp2 <- restaurants_join_rank[,c(4,5)]
tp <- tp1 * tp2; names(tp) <- paste("r", names(tp), sep="")
restaurants_join_rank <- data.frame(restaurants_join_rank,tp)
print(restaurants_join_rank[c(1,17,35,36),c(1,6,7,8)], row.names = FALSE)
```

```
##                  userid             businessid rtype1 rtype2
##  4jGnjQUArjFKOnxShpGLXw Ya0oNqP8mzPU5HwMdgf5zQ    0.0  0.000
##  4jGnjQUArjFKOnxShpGLXw cmGR1HS9ms233roSllcglw    1.0 -0.250
##  8fApIAMHn2MZJFUiCQto5Q 2X5G4Ujq0s4Wfn4TC7gX0g   -0.5  0.000
##  8fApIAMHn2MZJFUiCQto5Q nVyL22lcEZoUNoAMgfKOyQ    0.0 -0.625
```

Workflow

Here are the steps that are done to be able to answer where I should eat tonight.

1. Uniform the data. Done by a Scala object.

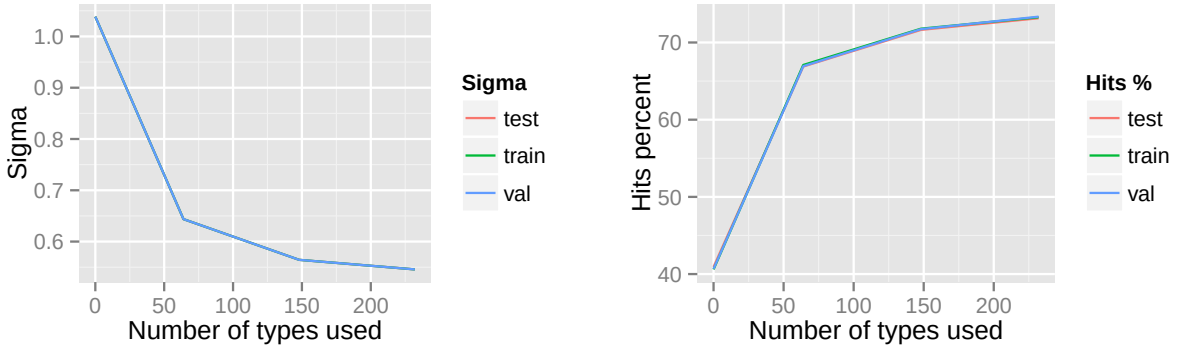
2. Import the data into R
3. Create the join containing business, review, user and business average stars. Also filtering out the one that is not a restaurant.
4. Use one of the types when there are many types that are highly correlated (higher or equal to 0.85).
5. Estimate the types of restaurant a user likes and dislikes and join it with the data set.
6. Filter out the data that are of interest. In this case I filtered out reviews that was dated before 1 January 2014.
7. Split the data into a training set, a validation set and a test set.
8. Create the fitting linear model.
9. Validate that the result is good enough. If not do some lm improvement, like removing the features that are not enough significant.
10. Use the model to find the best restaurants in the neighborhood.

Prediction model

After some minor investigations I decided to use the normal fitting linear model. Most of the features are either 0 or 1 or a number between 1 and 5 so it should be fitting good enough to use a linear model.

The features I used to predict the review star was the users' star average, the restaurant star average and the aggregated data I mentioned above regarding categories and attributes. Totally 106 categories and 126 attributes were used as a part of the final model.

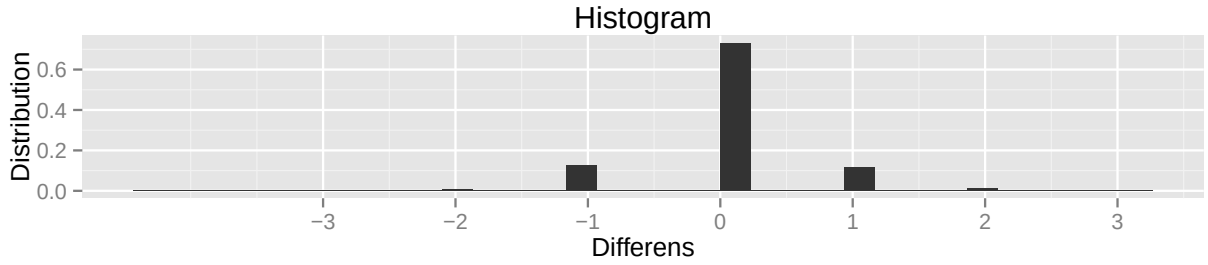
This is the result of comparing some alternatives regarding number of types:



As you may see in the plot the more types the better. There is also no bigger differences between the different sets. Without using any type as a feature there is a very poor result but by increasing number of types we get better and better results.

Results

The result was rather good. I decided to use the case where we have 232 types. The residual standard error was 0.5456806 and 73% of the predictions matches. Here is a histogram that shows the diff between the prediction and the real number of starts for the test data.



Here am I also going to answer my question: Where should I eat tonight? If a user is in central Las Vegas (latitude 36.114647 and longitude -115.172813) which restaurant should the user visit? It should be not more than 500 meters away. I compared two different users ("–_L4WuJxAfQkRtC1e43hg" and "Kq8-FUG7d_MT2qRNiNBJnA") and we see that there top 3 restaurants differs. There are only one in common.

```
lat <- 36.114647; lng <- -115.172813; max_dist <- 500
user <- user_join[user_join$urm_user_id=="--_L4WuJxAfQkRtC1e43hg",]
print(get_matching_companies(business, lat, lng, max_dist, user, lm_res), row.names = FALSE)
```

```
##      pred                     b_name b_stars
##  4.572379 Off The Strip Bistro & Bar     4.5
##  4.479236                  Guy Savoy     4.5
##  4.469623                    Picasso     4.5
```

```
user <- user_join[user_join$urm_user_id=="Kq8-FUG7d_MT2qRNiNBJnA",]
print(get_matching_companies(business, lat, lng, max_dist, user, lm_res), row.names = FALSE)
```

```
##      pred                     b_name b_stars
##  4.902528 Off The Strip Bistro & Bar     4.5
##  4.765725                Border Grill     4.5
##  4.557301     Chipotle Mexican Grill     3.5
```

Especially the second user, Kq8-FUG7d_MT2qRNiNBJnA, has the restaurant Chipotle Mexican Grill in the top 3 even if its average stars is only 3.5.

Discussion

I think the results are rather good. The prediction part is acceptable. With such a good prediction the top ranked restaurant in the neighborhood will probably at least be one of the top three.

Even if the result is rather good it is certainly possible to reduce the standard deviation. Since the time for producing the result was rather short many things are not yet investigated. Things that may be of interest are the texts for the reviews and also maybe for the tips as well, also a user's friend's reviews may be of interest. I am also using a normal lm. Maybe some of the Caret training models could be of interest.

It was a very interesting capstone and even if there was not time enough to get deeply into the problem it gives a god start and there are possibilities in improving the result.

All the code may be found on my github: https://github.com/e93fem/Data-Science-Capstone.