"Eat, Rate, Love" – A Proposal for Modifying Yelp's Rating Systems
1/12/2016

1. Introduction

According to the Atlantic magazine, the very first review submitted to Yelp was four stars for Truly Mediterranean in 2004, consisting of the 4 words "dirt cheap, good falafels."[1] Today, those 4 words have grown to over 2.2 billion words that would fill 16,894 Zagat guides and take 42 years to read out loud. Over 142 million unique visitors use Yelp every month (79 million of them through their mobile phones), accessing over 90 million reviews of over 2.1 million businesses. Every minute, 26,380 reviews are posted. Every day, 24,000 new photos are uploaded.[2]

This large proliferation of information can lead to one of the few weaknesses of using Yelp to decide "where to go Friday night" – with so many restaurant reviews, it can be challenging to figure out where to go to if there are a lot of restaurants of the same type in the same area, all with comparable rankings. A quick check of Indian restaurants in the Schaumburg, Illinois area, for example, shows 12 Indian restaurants, all with a rating of 3 to 4 stars. It would be useful if there were a way to help "cut through the clutter" and see if certain restaurants might stand out using different metrics. This analysis investigates the feasibility of 2 potential ways to do this.

The first is to create a new rating which gives more weight to those who have reviewed more restaurants of the same cuisine. Going back to the Schaumburg example, if someone has attended and reviewed all 12 restaurants, then their opinion should be given significantly more weight. Even someone who has been to 2 out of the 12 should be given more weight than someone who has just attended one.

The second proposed method is to create an "immigrant" rating. Going back to the Indian restaurant example, one characteristic of Schaumburg is that it has a lot of immigrant Indian workers working there temporarily for various tech companies. On the theory that those workers would actively seek out restaurants that remind them closest of "home cooking" and also that they tend to seek out places offering the most value, one thing people might do is check the ratings given by those with clearly Indian names to see what they think. The proposal would be to check the user name in Yelp to guess at who might be an "immigrant", and create a different rating for a particular ethnic cuisine given specifically by those users. This method admittedly has some clear deficiencies – it will ignore any "immigrants" who do not use their real names, and it will also mark as "immigrants" those who simply like an Indian name and choose to use it for their

[1] "Infographic: The Incredible Six-Year History of Yelp Reviews", Nicholas Jackson, The Atlantic, 7/20/11, http://www.theatlantic.com/technology/archive/2011/07/infographic-the-incredible-six-year-history-of-yelp-reviews/242072/

[2] Statistics all from "By the Numbers: 50 Amazing Yelp Statistics", http://expandedramblings.com/index.php/yelp-statistics/1/

Yelp ID. The hope is that there is enough information that cuts through the noise of those deficiencies to be able to provide useful information.

This analysis takes a dataset of Yelp data and analyzes the effect of these proposed changes. For each proposal, we’ll first do some exploratory analysis of the dataset to determine the feasibility of the change, and then we’ll try applying the change to see what effect it has on ratings -- focusing on the results in one city. Conclusions, recommendations, and potential next steps are then given in the end.

Data from the Yelp data challenge is used. This data consists of 1.6 million reviews from 366,000 users for 61,000 businesses in 10 cities: Edinburgh, Karlsruhe, Montreal, Waterloo, Pittsburgh, Charlotte, Urbana-Champaign, Phoenix, Las Vegas, and Madison.

2. The Yelp Dataset

The Yelp dataset consists of 5 files:

```
yelp_academic_dataset_business.json
yelp_academic_dataset_checkin.json
yelp_academic_dataset_review.json
yelp_academic_dataset_tip.json
yelp_academic_dataset_user.json
```

The main file is “review”. This consists of the text and star rating of each user review. The user and business are identified through a unique user_id and business_id. The date of the review as well as meta information on the # of votes the review got for being funny, useful, and cool, are also included here.

More details on the business can be found in the “business” file. Through the business_id, the full name of the business is given, along with its address, hours, coordinates, # of reviews, and average star rating (rounded to half a star). There is also a categories field for each business. It is this field that contains the cuisine (can have multiple values) as well as other general categories such as “Kids Activities,” “Active Life”, and “Night Life”. Finally, there is also an attributes field which contains meta information such as whether Take-out is supported, whether it has WiFi, and whether it can be classified as “romantic”, “classy”, “hipster”, “divey”, “touristy” etc.

More details on each user can be found in the “user” file. Through user_id, the user name is given, along with how long they’ve been in Yelp, the total # of funny/useful/cool votes they’ve received, the total # reviews they’ve done, and a review_id for each review they have done.

The “tip” file contains the text for tips left by each user, linked by user_id and business_id.

The “checkin” file contains check-in time information for 45,000 of the businesses.

As can be seen by the file extensions, all of the data is in JSON format.

3. Initial Data Wrangling

The very first step after downloading and unzipping the dataset was to find a program to split the files so that the very large files (specifically the “review” file, which is 1.3GB unzipped) could be looked at with a text editor and examined. The file “hjsplit” was used in Windows for this.

After doing this and getting a sense of what was contained in the files, it was determined that the following information was needed:

-From the “review” file: business_id, user_id, stars

-From the “user” file: user_id, name

-From the “business” file: business_id, business_name, city, categories, review_count, stars

“categories” is needed to extract the cuisine. “review_count” and “stars” are useful as sanity checks on the data as it is being analyzed (later, we will calculate the star rating ourselves and compare with the “stars” value given here as a quick check that all is kosher).

The “tips” and “check-in” files were not needed.

Given that a 1.3GB file is too large to read into R as-is (and would take a prohibitively long time even if it could), the first step was to preprocess the 3 files to extract only the fields needed. A python script was written to take care of this. Since python also has CSV support, conversion from JSON to CSV was done in this script as well.

In the python code there were only 2 slightly tricky things that were needed. The first was including “lineterminator = ‘\n’” in the ouputWriter() command when writing CSV. This was needed to prevent outputWriter() from inserting a blank line after each CSV entry. The second is the use of “.encode” to take care of UTF-8 characters that appear sometimes in the business name and city in the “business” file (used for accented characters with French names). These special commands are needed to prevent an error from happening in the CSV “outputWriter.writerow” command.

Running the python script greatly reduced the file sizes (“review” went from 1.3GB to a much more manageable 92MB). They were then ready to read into R.

Reading into R was done using the "read.csv" command:

```
reviews <- read.csv("yelp_academic_dataset_review.csv", header = FALSE)
users   <- read.csv("yelp_academic_dataset_user.csv", header = FALSE)
businesses <- read.csv("yelp_academic_dataset_business.csv", header = FALSE)
```

They were then combined into 1 data frame using the "inner_join" command so that business name, user name, city, cuisine, and rating are all in one data frame:

```
ru  <- inner_join(reviews, users)
rub <- inner_join(ru, businesses)
```

4. Analysis of Method 1: Giving More Weight to Multiple Reviewers of a Cuisine

The first step was to analyze the dataset to find the number of multiple reviewers there are for a cuisine – if there are very few, then adding weight to their opinions may ultimately have little impact on the overall rating. We also want to see whether the # of multiple reviews might vary from cuisine to cuisine (our deep dive will later focus on Indian cuisine – we first want to make sure that usage patterns are similar enough across cuisines to be able to generalize).

Let us first look at Indian cuisine. We add an "is_indian" column to the table based on whether the word "Indian" appears in "categories", using grepl:

```
rub$is_indian <- grepl("Indian", rub$categories) == TRUE
```

We then use subset to create a data frame of just reviews for Indian restaurants:

```
indian <- subset(rub, is_indian == TRUE)
```

Once we have this, we use the select, group_by, and summarise commands from dplyr to create a table of the # of reviews of Indian restaurants each user has done:

```
num_reviews_Indian <- indian %>% select(user_id, user_name, is_indian) %>%
      group_by(user_id) %>%
      summarise(tot_rev = sum(is_indian))
```

After doing this, we can use table, count, and mean to get review statistics:

```
> table(num_reviews_Indian$tot_rev)

   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   21   24   27
7814 1048  345  137   70   47   24   18    8   11    1    4    5    2    1    4    2    1    1    1    1    1
  30   93
   2    1
> count(num_reviews_Indian)
Source: local data frame [1 x 1]

      n
  (int)
1  9549
> mean(num_reviews_Indian$tot_rev)
[1] 1.376689
```

This yields a result of 9,549 total reviewers, with 7,814 doing just one review, 1,048 doing 2 reviews, and the rest doing 3 or more. The highest number of reviews is 93.

Roughly 10% of the users have done multiple reviews of Indian cuisine. This seems to imply there are enough users in general to justify trying a multiple review system (and not so many that the effect wouldn't be seen).

Using similar commands to the above, we can apply the same analysis to other cuisines. Let's use the top 10 "most-craved" ethnic cuisines in America, as ranked by Parade Magazine in an article on 5/15/2015[3]. Running similar commands on the Yelp dataset for these cuisines, we get the following result:

Cuisine	Total Reviewers	# > 1 Review	% > 1 Review	Avg Reviews per person	Max Reviews
(1) Chinese	33,359	7,733	23%	1.56	212
(2) Mexican	54,138	14,828	27%	1.77	145
(3) Italian	51,245	12,606	25%	1.63	90
(4) Japanese	44,849	11,688	26%	1.67	66
(5) Greek	10,820	1,799	17%	1.29	19
(6) French	19,127	3,413	18%	1.32	40
(7) Thai	20,699	4,032	20%	1.40	74
(8) Spanish	12,736	1,804	14%	1.25	27
(9) Indian	9,549	1,735	18%	1.38	93
(10) Mediterranean	19,400	3,534	18%	1.35	30

One can see that all cuisines had at least 10% of reviewers give multiple reviews. The average # of reviews also seems to go down as the # of total reviewers decreases. This makes sense – if there

[3] "Top 10 Ethnic Cuisines Americans Crave Most", Parade Magazine, 5/15/2015, http://parade.com/397203/parade/top-10-ethnic-cuisines-americans-crave-most/

are fewer restaurants of a cuisine in a city (which we'll assume correlates with the # of reviewers of that cuisine in a city), then there will be less chance of multiple reviews happening.

An interesting side note is that the total # of reviewers doesn't seem to exactly match up with the Parade rankings. For example, even though Chinese food is rated #1, there are more Yelp reviewers for Mexican, Italian, and Japanese food. Parade's definition of what Americans "crave" most appears to be slightly different from what the Yelp data indicates.

Given that all 10 cuisines have a rate of at least 10% multiple reviews (and in some cases as high as 25%), we conclude that it is worth proceeding further.

We then come to the real test -- modifying the rating using these weights and seeing what impact they have. Let's try this on Indian restaurants. We have the # of reviews for each user in num_reviews_Indian. If we join this back to our "indian" data frame containing all the individual ratings, we have a new table which has the rating the user gave as well as the # of Indian restaurants they have reviewed. Let's store this in a variable called cuisine-indian (or "cin" for short):

```
cin <- inner_join(indian, num_reviews_Indian)
```

Let's create a new weighted rank which is simply the # of stars times the # of reviews:

```
cin$weighted_stars <- cin$stars * cin$tot_rev
```

Now, we can once again use the group_by and summarise commands in dplyr to generate both the original average star rating (as a sanity check) and the new, weighted star rating:

```
new_rating_Indian <- cin %>% select(city, business_name, avg_stars, stars,
                                    tot_rev, weighted_stars) %>%
                             group_by(city, business_name, avg_stars) %>%
                             summarise(cnt = n(),
                                       avg = sum(stars) / cnt,
                                       new = sum(weighted_stars)/sum(tot_rev),
                                       dif = new - avg)
```

After doing this, we can use summary to get a sense of the effect of the new rating:

```
> summary(new_rating_Indian$dif)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.38900 -0.23490 -0.06903 -0.07623  0.05728  1.26900
```

We see that the new weights can move the rating down by as many as 1.4 stars or up as high as 1.3 stars, with usually there being relatively little effect.

If use view(new_rating_Indian) and sort based on the difference to see the restaurants that benefited the most, we see that quite a few of them have relatively few rankings:

	city	business_name	avg_stars	cnt	avg	new	dif
372	Phoenix	Copper Kettle Curry House	2.5	5	2.600000	3.868687	1.26868687
371	Edinburgh	Passage to India	3.5	5	4.000000	4.852459	0.85245902
370	Chandler	Biryani Kitchens	3.5	3	3.666667	4.500000	0.83333333
369	Gilbert	Kabob n Kurry Restaurant	3.5	44	3.727273	4.494505	0.76723277
368	Edinburgh	Namaste Kathmandu	3.5	10	3.700000	4.394366	0.69436620
367	Edinburgh	Shamoli Thai & Indian Cuisine	2.5	8	2.125000	2.733333	0.60833333
366	Tempe	Kabab Palace	3.5	94	3.702128	4.292439	0.59031171
365	Montreal	Restaurant La Nouvelle Lune Indienne	3.0	6	3.000000	3.578947	0.57894737
364	Montral	Bombay Tandoori	2.0	5	2.200000	2.777778	0.57777778
363	Charlotte	Udipi Indian Cuisine	3.0	5	3.200000	3.760000	0.56000000

The top 3 beneficiaries, for example, all had 5 or fewer rankings. Our proposed tweak would have a disproportionately large effect in those cases. To remove those cases, let's only look at restaurants with more than 5 reviews:

```
nri5 <- subset(new_rating_Indian, cnt > 5)
```

Here if we repeat summary, we get a new smaller range:

```
> summary(nri5$dif)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.14000 -0.24060 -0.09318 -0.09156  0.05591  0.76720
```

We can see that the impact is an increase of up to 0.77 stars and a decrease of as much as 1.14 stars. The number of restaurants goes down from 372 to 270.

The most important question, though, is would this difference potentially help a user get better results from Yelp? Let's look at the sample case of Indian restaurants in Tempe, Arizona. The results for Indian restaurants with more than 5 reviews are as follows:

	city	business_name	avg_stars	cnt	avg	new	dif
243	Tempe	Aachi Southindian Kitchen	3.5	28	3.678571	3.353293	-0.3252780154
244	Tempe	Bombay Palace	4.0	6	3.833333	3.625000	-0.2083333333
245	Tempe	Chutney's Indian Cuisine	3.5	121	3.702479	3.419048	-0.2834317198
246	Tempe	Copper Kettle-Salads Balti & Taandoori Grill	3.0	11	3.181818	3.715556	0.5337373737
247	Tempe	Curry Corner	4.0	209	3.746411	3.460808	-0.2856038823
248	Tempe	Delhi Palace Cuisine of India	4.0	111	3.747748	3.640167	-0.1075803837
249	Tempe	India Grill	4.0	62	3.870968	3.843621	-0.0273463428
250	Tempe	Kabab Palace	3.5	94	3.702128	4.292439	0.5903117128
251	Tempe	Kohinoor Cuisine of India	3.5	63	3.523810	3.432353	-0.0914565826
252	Tempe	Little India	4.0	50	4.060000	4.257812	0.1978125000
253	Tempe	Nandini Indian Cuisine	4.5	102	4.539216	4.470000	-0.0692156863
254	Tempe	Pasand	3.0	34	3.117647	3.328267	0.2106204184
255	Tempe	Passage To India	3.5	76	3.763158	3.865753	0.1025955299
256	Tempe	Priya Indian Cuisine	3.0	17	3.176471	3.059072	-0.1173988583
257	Tempe	Royal Taj	3.5	99	3.757576	3.750769	-0.0068065268
258	Tempe	Southern Spice	3.5	25	3.440000	3.200000	-0.2400000000
259	Tempe	Taj Mahal Indian Cuisine	3.5	66	3.757576	3.162393	-0.5951825952
260	Tempe	Tasty Kabob	4.0	80	4.000000	3.753012	-0.2469879518
261	Tempe	The Dhaba	4.0	274	4.062044	3.974551	-0.0874928974
262	Tempe	Udupi Indian Veg & Vegan Cuisine	4.0	141	3.957447	3.344196	-0.6132512892

From the ratings, one can see 8 restaurants with an official rating of 4 (Bombay Palace, Curry Corner, Delhi Palace, India Grill, Little India, Tasty Kabob, The Dhaba, and Udupi Indian) and 1 with a 4.5 (Nandini). Looking at the new weighted rating we get:

	Official	Old	New
Bombay Palace	4	3.83	3.63
Curry Corner	4	3.75	3.46
Delhi Palace	4	3.75	3.64
India Grill	4	3.87	3.84
Little India	4	4.06	4.26
Tasty Kabob	4	4.00	3.75
The Dhaba	4	4.06	3.97
Udupi Indian	4	3.96	3.34
Nandini	4.5	4.54	4.47

One can see that, with the weighted rating, “Little India” gets just enough of a boost to round it up to 4.5 stars, so in this case the weighted rating system would offer a good initial choice among the 8 options. Based on this example from real data, one can that a “weighted rating” that takes into account the # of other reviews a user has done of a cuisine could be potentially useful.

5. Analysis of Method 2: Adding an "Immigrant" Rating

The second proposed tweak is to add an "immigrant rating." This section tries this on the Yelp database for Indian cuisine and analyzes the results.

Since the Yelp database does not include ethnicity in its user database, the only way to try to guess ethnicity is through the user name. The first step was to generate a list of names that would qualify as being uniquely Indian. To do this, the user names in the Yelp "user" dataset were examined for potential candidates. When a name looked like a potential candidate, the question "Is ___ an Indian name" was googled to see if sites such as www.indiaparenting.com, www.modernindianbabynames.com, and www.indiachildnames.com would show up in the top of the search result. Anything that did so was included in the list. Any popular names in America were left out ("Daniel", for example, appears in the indiaparenting.com database but is left out). This resulted in 608 names, ranging from "Aayush" and "Abhijeet" to "Yogesh" and "Yuvaraj". The list of names used is included in Appendix A of this document.

This list of names was read into R using the scan command:

```
inames <- scan("indian_names.txt", what = character())
```

Taking the "indian" dataset from Method 1 that contains the business name, reviewer name, and star rating for all Indian restaurants in the dataset, we add a "reviewer_indian_name" field by using %in%:

```
indian$reviewer_indian_name <- indian$user_name %in% inames
```

Also, to simplify calculation later, we add an "istars" field which is simply the number of stars the reviewer gave the restaurant if that person has a uniquely Indian name, and a 0 otherwise:

```
indian$istars <- indian$stars * indian$reviewer_indian_name
```

After doing this, we can find out how many reviews fall under the "immigrant" category by using the "table command":

```
> table(indian$reviewer_indian_name)

FALSE  TRUE
11872  1274
```

We can see that, out of the 13,146 reviews, 1,274, or a little under 10%, of the reviews are eligible for the "immigrant" rating. This seems like a substantial enough amount to go forward.

Using “group_by” and “summarise” from dplyr, we can regenerate the average rating for a restaurant (as a sanity check) and also the “immigrant” rating, which would simply be the sum of the newly generated “istars” for each restaurant divided by the # of “immigrants” who reviewed that restaurant (found by “sum(reviewer_indian_name)” for each group):

```
avg_rating_Indian <- indian %>% select(business_id, business_name, city, stars,
                                       avg_stars, reviewer_indian_name,
                                       is_indian, istars) %>%
                                group_by(city, business_name, avg_stars) %>%
                                summarise(count = n(),
                                          nin = sum(reviewer_indian_name),
                                          pin = sum(reviewer_indian_name) / n(),
                                          avg = sum(stars) / count,
                                          ias = sum(istars) / nin,
                                          dif = ias - avg)
```

Here, “nin” stands for “number of Indian names”, “pin” stands for “percentage with Indian names”, “ias” stands for “Indian (rating) average stars (the new rating)”, and “dif” stands for the difference between the new and the original rating. “avg_stars” from the Yelp database is the official average stars rating recorded for that business.

After doing this and sorting the results by the difference between the new rating and the original rating, we see that there are dramatic differences, but in many of those cases the difference is due to there being only 1 immigrant:

	city	business_name	avg_stars	count	nin	pin	avg	ias	dif
1	Edinburgh	Lancers Brasserie	4.0	7	1	0.14285714	3.857143	1.000000	-2.8571429
2	Pittsburgh	Indian Spices	3.5	34	1	0.02941176	3.764706	1.000000	-2.7647059
3	Edinburgh	Mezbaan South Indian Restaurant	3.5	13	1	0.07692308	3.692308	1.000000	-2.6923077
4	Edinburgh	Indian Cavalry Club	3.0	6	1	0.16666667	3.166667	1.000000	-2.1666667
5	Pittsburgh	Maharaja Restaurant	3.0	11	1	0.09090909	3.090909	1.000000	-2.0909091
6	Phoenix	India Palace	4.0	141	3	0.02127660	4.028369	2.000000	-2.0283688
7	Charlotte	Tamarind	3.0	3	1	0.33333333	3.000000	1.000000	-2.0000000
8	Las Vegas	Sai India Curry	3.0	11	3	0.27272727	3.000000	1.000000	-2.0000000
9	Las Vegas	Pyaar India Restaurant	3.0	21	1	0.04761905	2.952381	1.000000	-1.9523810
10	Las Vegas	Samosa Factory	4.0	136	7	0.05147059	4.066176	2.142857	-1.9233193
11	Madison	Taj Indian Restaurant	4.0	39	3	0.07692308	3.923077	2.000000	-1.9230769
12	Pittsburgh	Prince of India	3.0	32	1	0.03125000	2.906250	1.000000	-1.9062500

Showing 1 to 13 of 372 entries

Let’s set an arbitrary value of needing at least 5 “immigrants” to be able to generate a useful “immigrant rating”. We use “subset” to screen for this:

```
ari5 <- subset (avg_rating_Indian, nin > 5)
```

When we do this and look at the values with the greatest difference, the max difference decreases from -2.85 to -1.9:

	city	business_name	avg_stars	count	nin	pin	avg	ias	dif
1	Las Vegas	Samosa Factory	4.0	136	7	0.05147059	4.066176	2.142857	-1.923319328
2	Charlotte	Saffron Indian Cuisine	3.5	87	12	0.13793103	3.505747	2.000000	-1.505747126
3	Tempe	Delhi Palace Cuisine of India	4.0	111	9	0.08108108	3.747748	2.333333	-1.414414414
4	Charlotte	Woodlands	4.0	94	9	0.09574468	3.978723	2.666667	-1.312056738
5	Phoenix	Flavors of India	3.5	116	9	0.07758621	3.663793	2.444444	-1.219348659
6	Madison	Maharaja Restaurant	4.0	144	10	0.06944444	4.069444	2.900000	-1.169444444
7	Glendale	Tandoori Times 2 Indian Bistro	3.5	119	7	0.05882353	3.621849	2.571429	-1.050420168
8	Chandler	Woodlands Vegetarian South Indian Kitchen	4.0	97	13	0.13402062	4.103093	3.153846	-0.949246630
9	Charlotte	The Blue Taj	4.0	136	9	0.06617647	4.139706	3.222222	-0.917483660
10	Scottsdale	Indian Paradise	4.0	143	8	0.05594406	3.916084	3.000000	-0.916083916

Using summary we see that the potential difference ranges from -1.9 to an increase of .31, with the mean value being -0.4 (this is interesting – perhaps reviewers from India or of Indian descent tend to be harder on Indian restaurants than reviewers from America …)

```
> summary(ari5$dif)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.92300 -0.72230 -0.34500 -0.43600 -0.09828  0.30980
```

The other item to note is that, when imposing the requirement that there be at least 5 immigrant ratings, the number of restaurants with an "immigrant rating" decreases from 392 to 70.

Once again, as was the case when looking at method 1, the really important question is whether this rating has a potential to improve the user experience. Let's again look at the test case of Indian restaurants in Tempe, Arizona. Sorting by City and looking at Tempe, we get:

	city	business_name	avg_stars	count	nin	pin	avg	ias	dif
32	Tempe	Chutney's Indian Cuisine	3.5	121	30	0.24793388	3.702479	3.300000	-0.402479339
45	Tempe	Curry Corner	4.0	209	23	0.11004785	3.746411	3.565217	-0.181194092
3	Tempe	Delhi Palace Cuisine of India	4.0	111	9	0.08108108	3.747748	2.333333	-1.414414414
12	Tempe	India Grill	4.0	62	8	0.12903226	3.870968	3.000000	-0.870967742
15	Tempe	Kabab Palace	3.5	94	11	0.11702128	3.702128	2.909091	-0.793036750
18	Tempe	Kohinoor Cuisine of India	3.5	63	10	0.15873016	3.523810	2.800000	-0.723809524
65	Tempe	Little India	4.0	50	15	0.30000000	4.060000	4.133333	0.073333333
51	Tempe	Nandini Indian Cuisine	4.5	102	7	0.06862745	4.539216	4.428571	-0.110644258
59	Tempe	Passage To India	3.5	76	9	0.11842105	3.763158	3.777778	0.014619883
30	Tempe	The Dhaba	4.0	274	29	0.10583942	4.062044	3.655172	-0.406871382
17	Tempe	Udupi Indian Veg & Vegan Cuisine	4.0	141	14	0.09929078	3.957447	3.214286	-0.743161094

There are 6 restaurants with an official rating of 4.0 and 1 with a rating of 4.5. Let’s see what the new rating gives:

	Official	Old	New
Curry Corner	4	3.75	3.57
Delhi Palace	4	3.75	2.33
India Grill	4	3.87	3.00
Little India	4	4.06	4.13
The Dhaba	4	4.06	3.66
Udupi Indian	4	3.96	3.21
Nandini	4.5	4.54	4.43

With the “immigrant rating” it becomes easier to differentiate between the 7 choices. The first thing we see is that “Nandini” maintains its high 4.5 rating. That would be the clear first choice. Let’s say that you’ve already tried Nandini and want to choose between the six 4 star choices. Here with the new rating, we see dramatic drops for “Delhi Palace”, “India Grill”, and “Udupi Indian.” Of the 3 remaining choices, we see that “Little India” is the only choice that actually increased in value compared to the overall average (and would be the only one to round to 4 stars). In this case the new rating would result in a clear recommendation.

It is interesting to note that the recommended choice of Little India is the same one recommended by using Method 1. Perhaps there is some interrelatedness with more immigrants also being those that would be more likely to try different places.

It is also interesting to note that “Bombay Palace” and “Tasty Kabob”, which showed up among the Method 1 finalists, did not have enough “immigrants” attending to qualify them for a rating with Method 2.

Using this scenario as an example, the conclusion is that including an “immigrant rating” could be a useful data point in selecting between different places that would otherwise have the same rating.

6. Conclusions / Recommendations

The following conclusions can be made from this analysis:

(1) For the most popular cuisines, at least 10% of the reviewers have also reviewed another restaurant of the same cuisine. This number goes as high as 27% for Mexican food but steadily drops as the number of restaurants that have that cuisine in a certain town drops.

(2) For the Yelp academic dataset, a little less than 10% of the reviewers of Indian food had user names that are uniquely Indian.

(3) A weighted rating does appear to be useful in certain situations and can help winnow the field further when choosing between many restaurants with similar ratings. If we require at least 5 reviews to have such a rating, the # of Indian restaurants eligible for such a rating drops from 372 to 270. The impact ranges from a drop of -1.14 stars to an increase of .77 stars.

(4) An "immigrant rating" can also be potentially useful, though if there is a requirement of at least 5 "immigrants", then the # of eligible restaurants drops drastically from 372 to 70. The potential impact ranges from a drop of -1.92 stars to an increase of .31 stars.

Based on this, we make the following recommendations:

(1) "Sanity check" the restaurants recommended by these new ratings – hire a panel of 5 locals who are passionate about the cuisine in that city and get their opinion on whether the suggestions agree with what they think (does "Little India" in Tempe make sense as a choice to bring out, for example).

(2) If things check out, consider trying a test run of the new ranking in areas that have a large number of the same type of restaurant. For example:

-Chinese restaurants in the Bay Area
-Pho restaurants in the LA / San Diego Area
-Indian restaurants in the Chicago area.

(3) To increase the chance of acceptance of these rankings, consider keeping the recommendations positive – for example, rather than listing the raw "immigrant ranking",

just add an "authentic" badge to those restaurants that score better than 4 stars and are rated by more than 5 "immigrants" (or, for the weighted average, add a "Liked by experts" badge). This makes things easier for the user to process and helps avoid controversy / pushback from restaurants that would be hurt by giving a raw score.

6. Potential Next Steps

Besides the general suggestion of "sanity checking" the results by getting the opinion of human reviewers mentioned above, here are potential next steps for each of the methods:

Method 1:

(a) Experiment with capping the maximum multiplier to 10 (the current system would assign a multiplier of 212 to the one person who reviewed 212 different Chinese restaurants – which is way too large).

(b) Consider adding "cool" and "useful" vote counts to further modify the relative weights given to multiple reviewers.

Suggestion 2:

(a) Find a way to automate generation of the name lists. For example, generate it by scraping an Indian baby names site and then scrubbing it to remove any common American names.

(b) Try other ethnicities to see if there are enough ethnic names to use – Chinese, Vietnamese, and Middle Eastern might be good ones to try next.

(c) There may be something interesting in those restaurants that have a large percentage of "immigrant" reviewers. Perhaps this could be another criteria for an "authentic" badge.

7. Final Remarks

If the person that wrote "dirt cheap, good falafels" were to write a review of dining in America, they might write "good variety, hard to choose." Yelp is a great app to make that choice easier. Hopefully, with the addition of tweaks such as those suggested in this analysis, that choice can be made a little bit easier. Although making the stomach happy is often both literally and figuratively a matter of "the gut", there are times when a little data-driven analysis can also help make the stomach happy.

Appendix A: List of Indian Names Used

Aayush
Abhi
Abhijeet
Abhijit
Abhilash
Abhinandan
Abhinav
Abhinay
Abhisek
Abhishek
Abilash
Achyuthan
Aditi
Aditya
Adnan
Ahana
Aisha
Aishaa
Ajay
Ajinkya
Akash
Akhil
Akhilesh
Akshada
Akshay
Akshaya
Alina
Alok
Amal
Amalee
Amalia
Aman
Amandeep
Amar
Amaris
Amarnath
Ambika
Ami
Amil
Amir
Amirah
Amisha
Amit
Amogha
Amrita
Anagha
anand
Anand Kumar
Ananth
Ananya
Aneesh
Ani
Aniket
Anil
Anindya
Anirudh
Anish
Anju
Ankan
Ankit
Ankita
Ankur
Ankush
Anoop
Anshul
Anu
Anubhuti
Anuj
Anum
Anup
Anupama
AnuPriya
Anurag
AnurAg
Anush
Anusha
Apara
Aparna
Apeksha
Aravind
Aravind
Baalaaji
Archana
Arin
Arindam
Arjun
Arpita
Arshad
Arul
Arun
Arunkumar
Arvind
Asha
Ashank
Ashish
Ashmita
Ashok
Ashwin
Asish
Asmita
Asodha
Atreyee
Atul
Atulya
Avinash
Kumar
Avishek
Avishekh
Ayesha
Baban
Babjee
Bala
Balaji
Bharadwaj
Bharat
Bharath
Bhavana
Bhavik
Bhavin
Bhavisha
Bhavya
Bhrata
Bhujang
Bhumi
Bhumika
Bibek
Bijal
Brij
Brijen
Brijesh
Brinda
Chaitanya
Chaitra
Chandan
Chandini
Chandra
Chandrasekar
Chaundra
Chetan
Chetu
Chhayakanta
Chiku
Chintan
Chinu
Chirag
Chitra
Chitta
Cintya
Dakshina
Debashri
Deedar
Deeksha
Deepak
Deepali
Deepanjali
Deepesh
Deepika
Deepthy
Deepti
Deiva
Dev
Devang
Devesh
Dhanasekar
Dheeraj
Dhinakaran
Dhruv
Digvijay
Dinesh
Diplekha
Disha
Divyansh
Gaggandeep
Ganesh
Gaurang
Gaurav
Gautam
Gayatri
Geetha
Ghena
Girish
Gita
Gopi
Gurpreet
Gurpreet Singh
Hardeep
Haresh
Hariharan
Harinath
Harini
Harish
Harjit
Harpreet
Harsh
Harsha
Harshit
Himanshu
Hitesh
Humza
Jagadeesh
Jahan
Jai Veer Singh
Jaina
Jaswant A.
Jateen
Jatin
jaya
jayashri
jayatheerth
jaydei
JAYESH
jayshree
jeetendar
jharna
jimeet
jinesh
Kalpesh
Kalyan
Kamesh
Kanishk
Kannan
Karan
Karthik
Karthiknathan
Kartik
Karuna
Kashyap
Kavita
Kavitha

Appendix A: List of Indian Names Used

Kedar
Keerthi
Keren
Ketan
Kewal
Keya
Khushbu
Kira
Kiran
Kirra
Kirti
Kishore
Krishan
Krishna
KrishNa
Krishnasri
Kriti
Kritika
Krupesh
Kumar
Kunal
Kyara
Laks
Lakshmi
Lanka
Ksheera
Sagar
Laxmi
Madhu
Madhur
Madhuri
Mahadeva
Mahanth
Mahathi
Mahawish
Mahesh
Maithreyi
makku
Mala
Manav
Mandi
Mandisha
Mangala
Manikandan
Manish
Manjeera
Manveer
Mayur
Meena
Meera
meha
Mehta
Mehul
Mihir
Milind
Misbah
Mohun
Monal
Monali
Monish
Mukesh
Mukund
Munir
Nabeel
Nachiket
Nadeem
Naga
Nagdeep
Nahja
Najwan
Nakul
Namita
Namrata
Nanda
Nandini
Narayan
Naren
Naresh
Naveen
Navneet
Navpreet
Navya
Nayan
Neel
Neelam
Neelesh
Neeraj
Neha
Nidhi
Nikhil
Nilanjan
Nilesh
Nimish
Niraj
Niranjana
Nirmal
Nisarg
Nishant
Nishitaa
Nithesh
Nitin
Nitya
Nivedhitha
Padmaja
Padmashree
Palak
Pancham
Pankaj
Parikshith
Parth
Parthiv
Pavan
Pavi
Pavitha
Pavithra
Pavneet
Peeyush
Pinak
Piyoosh
Piyush
Pooja
Poorna
Poornima
Prbhakar
Prabhjot
Prabhu
Prabin
Pracheta
Prachi
Pragna
Prajod
Pranab
Pranathi
Pranav
Pranay
Praneeth
Praphul
Prasad
Prasanna
Prasen
Prashant
Prashanth
Prateek
Pratiba
Pratik
Pratiti
Praveen
Prayag
Preet
Preeti
Prem
Premsankar
Prerana
Priya
Priyam
Priyank
Prudhvi
Puneet
Punita
Purnima
Rachit
Raghav
Raghava
Raghu
Raghuram
Rahul
Rai
Raj
Raja
Rajat
Rajeev
Rajendra
Rjean K. C.
Rajesh
Rajiv
Rajni
Raju
Rakesh
Raki
Ram
Ramah
Raman
Ramandeep
Ramesh
Rasesh
Rashmi
Rav
Raveendran
Ravi
Ravi Krishna
Raviteja
Risha
Rishi
Rishik
Rishoo
Ritesh
Rohan
Rohit
Ronak
Roshan
Roshani
Roshni
Ruchi
Rupa
Rupu
Rushabh
Rushikesh
Sachin
Sadhu
Sagar
Sai
Sairam
Sakshi
Samara
Sambhav
Sameer
Samir
Sampad
Sandeep
Sandhya
Sangeetha
Sanjay
Sanjib
Sanjith
Sanjiv
Santosh
Sarang
Sarnnya
Saravana
Saravanan

Appendix A: List of Indian Names Used

Saroj
Sarun
Sashidhar
Satender
Sathya
Satya
Satyajit
Satyaswaroop
Satyin
Saumav
Saumya
Saurabh
Savinay
Sayed
Sayjul
Seema
Seema and Amit
Senthilkumar
Shailesh
Shaji
Shakerul
Shakil
Shalini
Shalu
Shamanth
Shanda
Shankar
Shantanu
Sharath
Shashank
Sheema
Sheetal
Shikha
Shilpa
Shilpashree
Shilpi
Shiraz
Shiv
Shiva
Shivani
Shivanshu
Shreejay
Shrikant
Shrimant
Shriya
Shruti
Siddharth
Sidhartha
Sidhu
Sirish
Sneha
Snigdha
Soham
Someshwar
Somnath
Sonal
Sonel
Soroush
Soujanya
Sourav
Souvir
Sowmiya
Spandana
Sree
Sri
Sri Balaji
Sridhar
Srikanth
Srikumar
Srini
Srinivas
Sriram
Srishti
Srivatsava
Subha
Subhash
Sdatta
Sudha
Sudhakar
Sudhanwa
Sudheer
Sudip
Sujith
Sukh
Suma
Sumaira
Sumaithri
Suman
Sumit
Sundar
Sunil
Supreeth
Surajit
Suraya
Surranna
Surya
Sushant
Suvir
Swapnil
Swati
Swikrit
Syed
Tanweer
Tapan
Tarun
Tripti
Tuhin
Tushaar
Tushar
Tushara
Umesh
Urvi
Urvish
Vaivelan
Vaibhav
Vaishali
Vaishnav
Vamsee
Vamsi
Varinder
Varun
Varuni
Vasista
Veena
Vemana
Venkat
Venkata
Venkesh
Vibhor
Vibhu
Vignesh
Vijay
Vijay Kumar
Vijaykumarty
Vijith
Vikas
Vikash
Vikram
Vinay
Vinayak
Vineet
Vineeth
Vinit
Vinod
Vinoth
Vinuth
Vipin
Vipin Das
Vipul
Vishal
Vishnu
Vishwa
Vivek
Yogesh
Yuvaraj