Trender: Using Location and Time to Discover Yelp Trends

John Loftin
Thomas Johnson

April 6, 2015

1 Introduction

1.1 Description

We wish to learn how location and time affects reviews.
For each state in the Yelp Dataset we will view how the number of reviews and ratings of businesses change over time.

1.2 Motivation

One of the partners on this project used to live in Champaign, IL, one of the cities in the Yelp dataset. Being Southern-bred, winter required a lot of heavy clothing. While walking around the University of Illinois campus, he would notice people in ice cream shops and frozen yogurt shops. The places would be full in the dead of the winter. This was confusing. Who eats ice cream in Winter? When the Yelp dataset was released, this led us to ask "How does climate affect consumer behavior?" This produced a related question that we actually hope to answer: How do date and location affect consumer behavior as shown by Yelp reviews.

2 Setup

2.1 Tables

We used SQL Developer to create our tables (see figure 1). Our tables were populated using SQL Loader, Python and OpenOffice Spreadsheet.

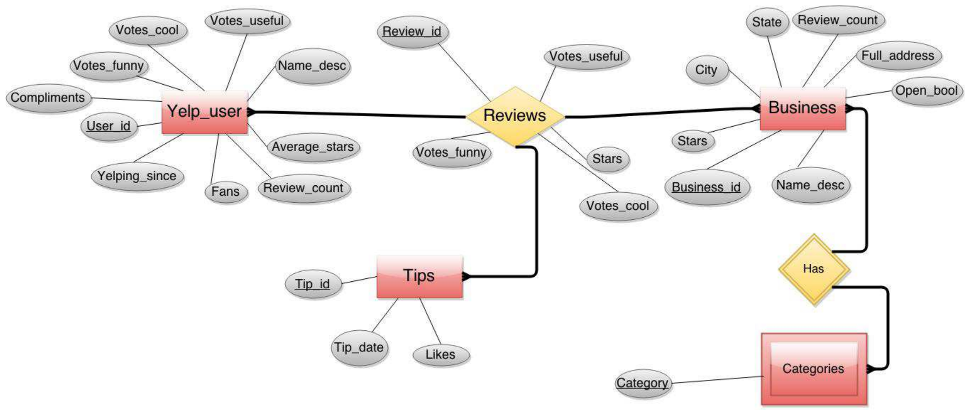

Figure 1: ER diagram.

We made the user id information from the dataset the primary key for the yelp_user table. The user_id from the dataset is encrypted and consists of letters and numbers, so we made this attribute of type varchar. We did this because it was simpler than generating numbers for the primary key. The compliments attribute on the ER diagram is actually a set of attributes (e.g. compliments_cool) that we left off for brevity. Although the vote attributes seem like redundant information which could be retrieved through an aggregate query on the review table, they are not. This is because reviews are provided only for a few cities, and the vote attributes in the user table are for all of the reviews of a user, in any city. To populate this table, we had to clean up the dataset and remove the friends column from the data, which gives the friends of a user as a list of user_ids. Since all of these ids are encrypted, they are fairly long strings, and could not be contained as of field of type varchar2(1000). We decided it was best to remove this column due to its size and the fact that it was not necessary for our project. Six entries from the dataset could not be loaded, as they had null user id values.

We made business id information from the dataset the primary key of the business table. Like the user_id, this business id is an encrypted string. The review_count column is not redundant information, since the number of reviews for each business in the review table is close to but not equal to review_count.

We made review id information from the dataset the primary key of the review table. Like the user and business ids, this information is an encrypted string. We made business_id a foreign key. We would have done the same for user_id, but obtained many foreign key violations as we tried to populate the table. We believe the review dataset contains reviews from users not in the yelp_user table. We formatted the dates on this table to the SQL date datatype using a Python program we wrote named date.py.

We created the categories table using information from the business dataset. When we initially populated the business table,there was a categories column, which contained a list of categories for each business (e.g. food, etc.). We wrote a Python program named categories.py to create the categories table. We made category and business_id the primary key for convenience.

The tips table was created using the tips dataset. We removed the text column from the dataset because it was too long as well as unnecessary, and replaced it with id numbers using a Python program we wrote named tips.py. The dates were formatted using date.py.

2.2 Python Program

The program sits on top of the Yelp schema we created. The program allows user to create a filter that we drag through the Yelp dataset. We allow users to pick two states of interest, a list of categories, a start date, an end date, and finally how to subdivide the time between those two dates. After applying the filter, we get statistics about the data including the mean and a review density (#reviews in subintervial/#reviews in entire interval).

2.3 Sample Queries

The first query examines how review ratings change with each month. This is useful for examining how review ratings change with seasons. The standard deviation given is the standard deviation of the mean.

```
select b.state, avg(r.stars), stddev(r.stars) / sqrt(count(r.review_id)) as std_dev,
count(r.review_id) as review_count,
extract(month from r.review_date) as Month, extract(year from r.review_date)
as Year
from review r join business b on r.business_id = b.business_id
where (r.review_date between '01-Jan-10' and '31-Dec-10') and (b.state =
```

'AZ' or b.state = 'IL')
group by b.state, to_char(r.review_date, 'Month'), extract(month from r.review_date), extract(year from r.review_date)
order by state, Year, Month;

The next query examines how review ratings change with the years. This is useful for examining if businesses are improving in quality. To get rid of the season as a variable, the month is held constant.

select b.state, avg(r.stars), stddev(r.stars) / sqrt(count(r.review_id)) as std_dev, count(r.review_id) as review_count, extract(month from r.review_date) as Month, extract(year from r.review_date) as Year from review r join business b on r.business_id = b.business_id where (extract(month from r.review_date) = 12) and (b.state = 'AZ' or b.state = 'IL') and (extract(year from r.review_date) ¿ 2005) group by b.state, to_char(r.review_date, 'Month'), extract(month from r.review_date), extract(year from r.review_date) order by state, Year, Month;

3 Data Analysis and Results

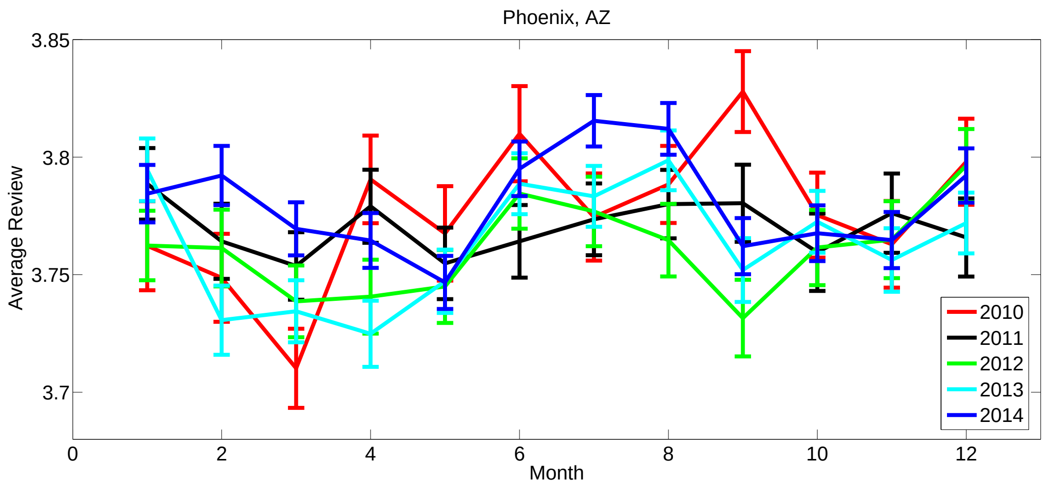

Figure 2: Variation of reviews with the month.

We will first examine how reviews change with the season. We will do this for Phoenix, Arizona and Urbana-Champaign, Illinois. All of our analysis was done in Matlab.

The average review rating for various years is given for Phoenix, AZ in figure 2. As you can see, ratings tend to decline from January to March, then increase until July, decline until October or November, then increase in December. The reviews tend to be higher in the summer, with the exception of December. The rating increase for December may be due to Christmas.

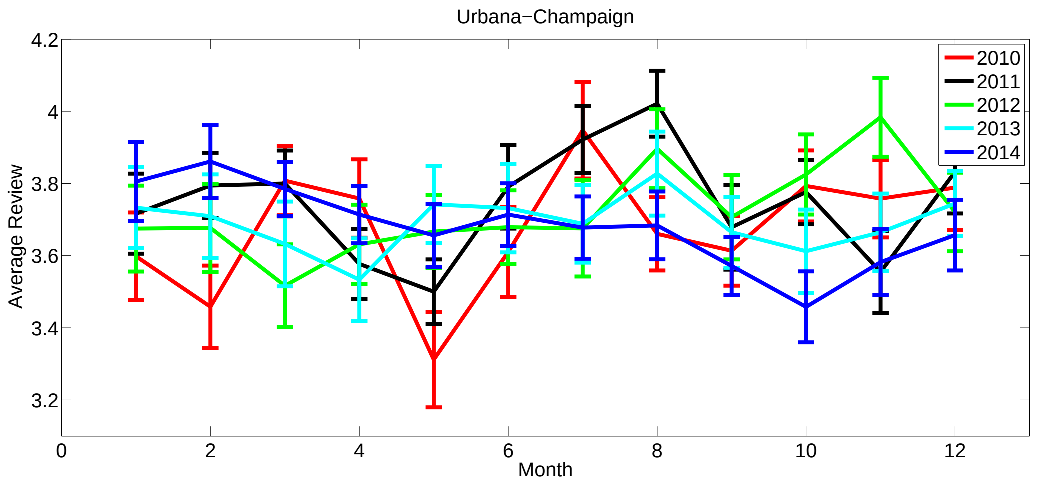

Figure 3: Variation of reviews with the month.

The average review rating for various years is given for Urbana-Champaign, IL in figure 3. As you can see, ratings generally decrease from January to May, then increase until August, decrease until October, then increase until December. However, these trends are not followed as closely as those for Phoenix, AZ. This may be because there is a significantly smaller sample size for Urbana-Champaign. Whereas there are thousands of reviews each month for Phoenix, there are around a hundred reviews each month for Urbana.
Next, we examine how reviews change with the year. We will hold the month constant (December) in order to eliminate season as a factor.

The average review from 2006 to 2014 for Phoenix, AZ is given by figure 4. A linear weighted fit is given, and the error bars correspond to one devia-

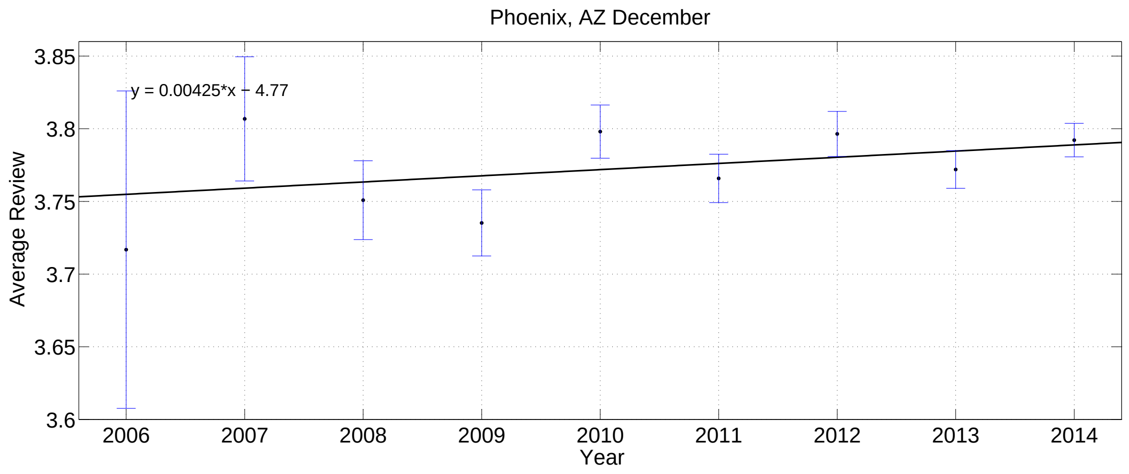

Figure 4: Variation of reviews with the year. A linear weighted fit is given.

tion. Since the line of best fit passes through 6 of the 9 error bars, the fit is a good one (statistically, the true mean should lie within one standard deviation 67% of the time). The review ratings have been increasing with time. There error bars are larger for earlier years, as there is a smaller sample size.

The average review from 2006 to 2014 for Urbana-Champaign, IL is given by figure 5. A weighted linear fit is given, and is a good fit as the line passes through 7 of the 9 error bars.

4 Conclusion

Using the Yelp database, we have discovered that reviews tend to be higher in the summer and in December. We have also discovered that reviews are increasing every year in Arizona, and declining in Urbana-Champaign. Possible extensions include adding support for more areas to filter than two areas, allowing to include subregions of cities instead of states. Finally, for data output, it would be nice to be able to output graphs and maps instead of just text output. More integration with the user data would provide more interesting analytical opportunities.

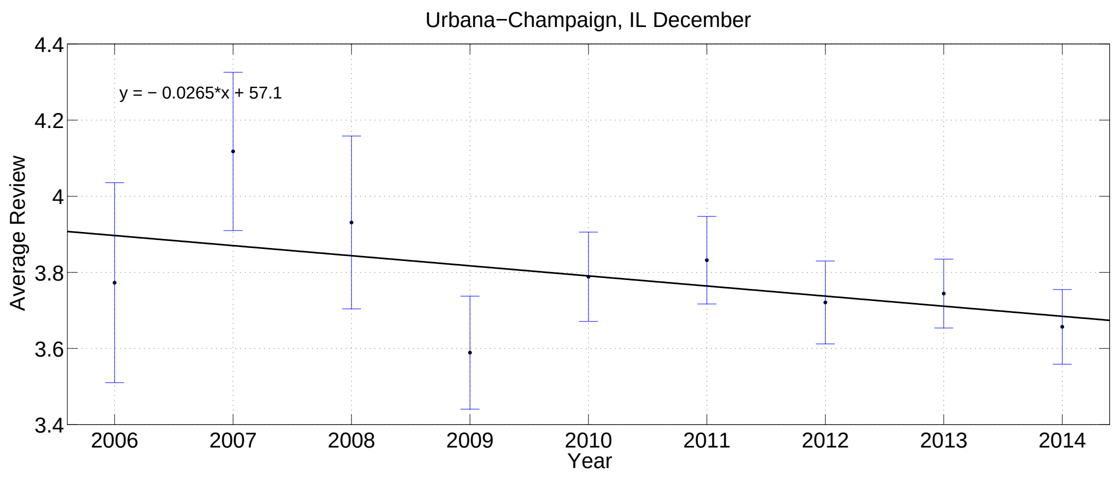

Figure 5: Variation of reviews with the year. A linear weighted fit is given.