Does a hotel's fate depend on user written reviews?

Using historical data from TripAdvisor to predict the popularity and attention that a hotel will gain

Mathew Teoh (301165239), Nathan Samchek (301110477)
12-1-2014
Simon Fraser University, CMPT 419, Group 12

Problem:

Finding a good hotel to stay at is much easier today than it was ten years ago. With sites such as TripAdvisor to provide firsthand accounts of hotel experiences, the future of a hotel depends on its customers more than ever. User written reviews could determine future customer experiences. For example, people reading about a "hotel's bad breakfast" may give a lower rating if they stay there. Even worse, they may not even stay at the hotel at all!

A similar study was conducted in "Inferring Future Business Attention" by Hood et al [1] from Carnegie Mellon University, where they used a dataset of Yelp reviews provided in the 2013 Yelp Dataset Challenge. It should be noted that due to lack of time and the sophistication of their methods, we did not follow their procedure exactly.

Our problem summarizes to the following: *given a hotel and its user written reviews, can we predict the amount of attention it will get, and its popularity in the future?* We will use terms developed during the course of this project whose meaning may not be clear to the reader. These terms will be **bolded** and explained.

Data Processing

Goal and general overview of processing

We obtained a TripAdvisor dataset provided by Wang et al [2] which contained over 12000 .json files, one for each hotel (link: http://times.cs.uiuc.edu/~wang296/Data/). (Although Wang et al conducted a study using the same dataset, their methods are more sophisticated than ours.) Each file contains information about the hotel itself, and a list of reviews. Each review contains information about the scores set by the user, the text they wrote, and the date they submitted it, ranging from May 2001 to May 2012.

Our goal is to set up a feature vector $\boldsymbol{x} \in \mathbb{R}^{800}$ that contains information about a given hotel (explained below) and a target vector $\boldsymbol{t} \in \mathbb{R}^3$ whose entries are the (future) average ratings, number of reviews, and average number of reviews per day, respectively. Given a hotel, both $\boldsymbol{x}$ and $\boldsymbol{t}$ depend on *disjoint subsets* of the hotel's reviews.

Given a hotel and one particular review, there is one particular date on which it was written. A value of $\boldsymbol{t}$ for a particular hotel must depend on the reviews with later review dates than those for $\boldsymbol{x}$. Additionally, across all the hotels, this dataset has an upper bound on the review dates (approximately May 2012). Thus, we do not have a continuous, real time set of TripAdvisor data off of which to base $\boldsymbol{t}$. This naturally led us to ask: "How do we split a set of hotel reviews to determine $\boldsymbol{x}$ and $\boldsymbol{t}$?" More specifically, we were concerned with determining a **cutoff date**, a fixed date universal across all hotels in the dataset, such that for any hotel, the reviews whose date occur *before* the cutoff date are used (*how* it's used is described in the sections below) to determine $\boldsymbol{x}$, and those that come after are used for $\boldsymbol{t}$. Given a continuous feed of real time TripAdvisor reviews, this cutoff date would be the current day (i.e. whatever "today" is). Methods on determining this cutoff date and constructing $\boldsymbol{x}$ and $\boldsymbol{t}$ vectors, are described below.

Structure of the data

To work with the dataset in MATLAB, we parsed the .json files using JSONlab, available on the MATLAB File Exchange [3]. The contents of the .json file data for each hotel can be arranged in the lists of elements below, where a nested list's elements are contained in one instance of its parent element.

- Hotel: *many in the dataset*
    - Hotel info: *1/hotel*
        - Price: *1/hotel*
        - Address: *1/hotel*
    - Reviews: *many per hotel*
        - Ratings: *1/review, out of 5 stars*
        - Text written by user: *1/review*
        - Date: *1/review*

While using a hotel's location may have affected its future popularity or attention, we decided to exclude it for sake of the simplicity of our model.

Cleaning raw data

Our top priority was to make sure that $\boldsymbol{x}$ and $\boldsymbol{t}$ contain complete information. This step of cleaning the data occurred *before* separating the data into training and testing groups because ensuring completed values for each of $\boldsymbol{x}$ and $\boldsymbol{t}$ is necessary for both groups. That is, "seeing" the test data for cleaning purposes does not give us an "unfair advantage" for training the model.

We first decided to remove any hotels with up to only 10 reviews, since we reasoned that splitting such sets (into $\boldsymbol{x}$ and $\boldsymbol{t}$) of reviews would result in subsets of reviews too small to give a reliable value for $\boldsymbol{x}$ or $\boldsymbol{t}$.

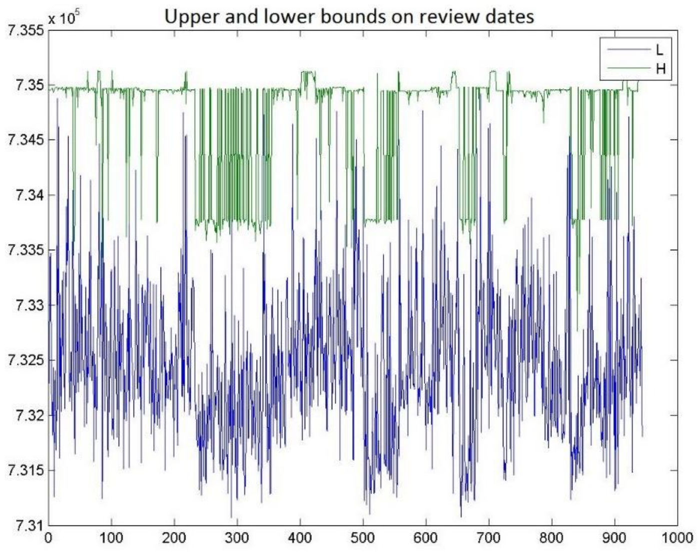

Figure 1

We reasoned that incomplete data would still arise even after ensuring a minimum number of reviews per hotel. Consider determining a cutoff date between $\boldsymbol{x}$ and $\boldsymbol{t}$. Figure 1 shows a plot of a subset of the total hotels.

The y-axis measures the review dates, using MATLAB's system of serial date numbers, which measures the number of days since January 0, 0000. Each integer value on the x-axis indexes a particular hotel in the subset above. The green line represents the latest (thus the highest serial number, or **upper bound**) review date across the sample of hotels, and the blue represents the earliest (thus the lowest serial number, or **lower bound**) review date.

Determining a cutoff date means drawing a horizontal line in Figure 1. To prevent "missing values" in $\boldsymbol{x}$ and $\boldsymbol{t}$, this would mean removing hotels whose earliest review date comes after the cutoff date, or hotels whose latest review date comes before the cutoff date. After looking at the upper and lower bounds of review dates for each hotel, we decided that in order to minimize the number of hotels lost, and to ensure that our data generated later on was complete, that acceptable hotels would have date ranges that contained a certain time interval. This meant upper bound dates greater than a serial number of 7.337×10^5 (19-Oct-2008) and lower bound dates less than 7.3365×10^5 (30-Aug-2008). (This decision was made manually, and one could argue that a better decision for reducing data loss could be made with a line sweeping algorithm, but we decided not to pursue it to save time.) After removing hotels who's upper and lower bound of dates fell within this **interval** (i.e. [30-Aug-2008 to 19-Oct-2008]), we obtain a modified plot, Figure 2.

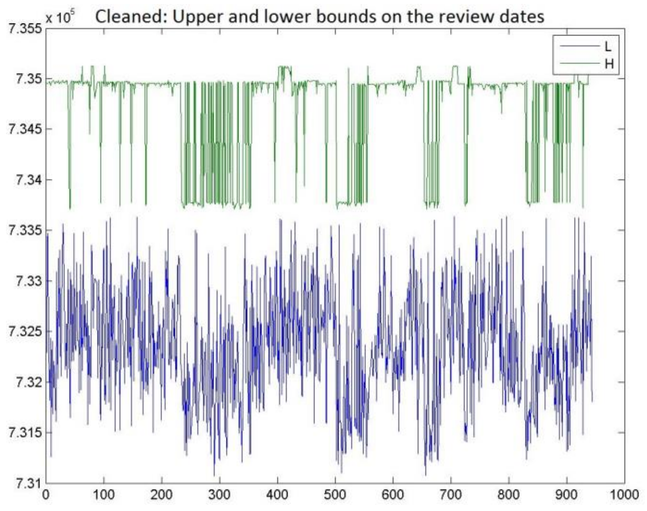

Figure 2

The set of hotels described in Figure 2 are the same as those in Figure 1. Figure 2 has a horizontal white band of space within which the cutoff date can be placed.

We also considered a set of training data whose hotel review dates satisfy a similar condition; rather than the upper and lower bounds of dates containing the time interval above, the first and third quartiles of the reviews must contain the interval to qualify as an acceptable hotel. This condition will be referred to as the **quartile constraint**. We decided that enforcing that quartile constraint would mean that the remaining hotels to train on was guaranteed to have more reviews per hotel. We were curious to see its effects on testing accuracy.

Finally, we discarded hotels that did not have information about its price. This was because we decided that price would be a useful feature to learn on, and missing prices would have resulted in incomplete data.

Splitting the data into training and testing

We chose to take 30% of the cleaned data as test data, which was 2515 hotels, not satisfying the quartile constraint. For the training data that did *not* satisfy the quartile constraint, this meant having around 5866 hotels, and 2788 hotels for the training data that satisfied the quartile constraint. This meant two training sets to train two models, and only one test set. Using the non-quartile constraint data to test both models can be justified because there is simply *more data to test against* without using the quartile constraint.

Parsing review text

The TripAdvisor dataset contains 316 keywords that describe various aspects of a hotel, such as cleanliness and service. We used all of these keywords assuming that they had a significant influence on the hotel's future.

To determine more key words to train our model on, we compiled over 900MB of user written reviews from the training data (no quartile constraint), and extracted a list of adjectives, adverbs, and gerunds occurring in the review texts with the Python Natural Language Toolkit (NLTK) to. We assumed that adjective and adverbs effectively describe the customer's experience, and gerunds often contain descriptive language, such as "welcoming"). We further assumed that adjectives were most effective, and took the 316 most occurring adjectives. We then took the same proportion (i.e. proportion of adjectives taken) of adverbs and gerunds, because we assumed that "fairness" in determining features involved taking the same proportion of adverbs and gerunds. This resulted in 791 keywords used to construct $\boldsymbol{x}$.

Splitting x from t

To minimize data loss in the construction of $\boldsymbol{x}$ and $\boldsymbol{t}$**,** we chose a cutoff date as the midpoint in our interval described in "Cleaning raw data" with serial number 733675, which corresponds to a date of 24-Sep-2008. It is important to note that, again, this was manually chosen, and that other more effective methods of determining a cutoff date may exist.

Generating features and targets

Given a certain hotel, the first 791 entries of $\boldsymbol{x}$ determine the number of times per review (i.e. **relative occurrence**) the corresponding keywords appear in that hotel's set of reviews. We measured this for each hotel by concatenating all of the text in reviews (in only that hotel) before the cutoff date. To measure relative occurrence for each keyword, we searched the concatenated text for that keyword, then dividing the number of times it occurred by the number of reviews (for that hotel, before the cutoff date).

The other 9 features were: average rating score; earliest, 1st quartile, median, 3rd quartile, and latest review dates; average price; and greatest number of days with no review for the hotel. All of these are measured for the hotel before the cutoff date. These features resulted from certain assumptions we made:

- Historical ratings of a hotel may affect future attitudes towards a hotel;
- Quartiles of review dates captures the time periods that the hotel received attention, which may predict future attention;
- Average price combined with the other features may determine the future popularity of the hotel (e.g. an expensive hotel with poor ratings may receive less attention in the future); and
- Longer time periods of no reviews may indicate that the hotel is failing to attract customers, which may affect future business.

Constructing $\boldsymbol{t}$ for each hotel involved averaging the ratings across all of the reviews (first entry). Measuring the "future attention" for a hotel involved counting the number of reviews (second entry). The third entry of $\boldsymbol{t}$ was the second entry divided by the number of days the reviews span. These reviews are the ones after the cutoff date. We assumed that higher ratings indicated a more popular hotel, and a larger number of reviews meant more attention for the hotel. The third entry was our attempt at making "attention" more of a relative measure, since if we had real time data, counting the number of reviews only makes sense if it's per day.

Method of Creating Models

Description of the models considered

We performed regularized, regression of the form:

$$[\mathbf{W}] = (\lambda[\mathbf{I}] + \boldsymbol{\Phi}^T(\boldsymbol{x})\boldsymbol{\Phi}(\boldsymbol{x}))^{-1}\boldsymbol{\Phi}^T(\boldsymbol{x})\boldsymbol{t},$$

with $\boldsymbol{x} \in \mathbb{R}^{800}$, $\boldsymbol{t} \in \mathbb{R}^3$, and design matrix $\boldsymbol{\Phi}$.

The basis functions we considered were polynomials of $\boldsymbol{x}$ with degrees ranging from 1 to 3. Higher degrees and other basis functions were not considered due to learning time constraints. We also considered regularization values $\lambda \in [0, 0.01, 0.1, 1,10,100,1000]$.

To determine the best basis function and λ we used 10-fold cross validation on our training data and chose the model that gave the lowest RMS error over all ten sets.

Using subsets of features/data

We were curious about how accurate various subsets of the features were at modelling the data.

We considered five subsets of features and compared the results of training/testing with respect to those subsets.

The subsets of features were:

1. All features
2. Numeric features only
3. Textual features only
4. Textual features from aspects only
5. Numeric features and textual features from NLTK analysis only

In descending order of size: 1, 3, 5, 4, 2

The first subset was chosen to have something to compare the accuracy of the other subsets against. The second and third subsets are disjoint and are used to compare the results of modelling with only numeric features (e.g. average rating, spread of dates, price, etc.) vs. the results of modelling with only textual features (i.e. how the reviewer is evaluating various aspects of the hotel/visit in the text of the review). The fourth and fifth subsets are also disjoint and are used to compare the results of modelling with only the "value" textual features (e.g. adjectives) and numerical features vs. the results of modelling with only the "aspect" textual features (e.g. room, location, service, etc.).

We were also curious about whether data with a smaller variance would provide a better model for prediction.

The two sets of data we considered were for when:

1. The pre-chosen cutoff date fell between the earliest and latest review dates for a hotel
2. The pre-chosen cutoff date fell between the first and third quartiles of the dates for a hotel (i.e. hotels that satisfy the quartile constraint).

It follows that set 2 is a subset of set 1.

Observations and Analysis

Results and Interpretations

Table 1, Table 2, and Table 3 show the resulting RMS, average percentage, and average absolute testing error. The blue values are for the data from set 1, and the green values are for the data from set 2. Minimum and maximum values for each of the target types are in bold.

RMS Error for Different Output Types and Selections of Features			
	Avg Ratings	# of Reviews	Avg # of Reviews per Day
All Features	**0.4920**	**139.786**	0.2069
Numeric Only	0.5127	140.733	0.2081
Text Only	0.5125	172.092	0.2256
"Aspect" Only	0.5425	175.110	0.2275
No Aspect	0.4937	146.714	0.2077
All Features	0.5174	151.969	**0.2050**
Numeric Only	0.5120	163.566	0.2146
Text Only	0.5308	176.760	0.2291
"Aspect" Only	**0.5529**	**177.970**	**0.2298**
No Aspect	0.5152	155.899	0.2134

Table 1

Additionally, Figure 3 shows a direct comparison of predicted values vs. target values for the average rating of future reviews when trained on set 1. The red line shows a perfect prediction, the blue data points are for data from set 1, and the green data points are for data from set 2.

Avg % Error for Different Output Types and Selections of Features			
	Avg Ratings	# of Reviews	Avg # of Reviews per Day
All Features	**11.94%**	418.2%	218.9%
Numeric Only	12.66%	412.5%	204.7%
Text Only	12.48%	480.6%	257.3%
"Aspect" Only	13.55%	434.4%	250.0%
No Aspect	12.09%	420.9%	213.2%
All Features	12.62%	421.7%	214.7%
Numeric Only	12.47%	**385.8%**	**122.6%**
Text Only	13.00%	**522.9%**	**305.1%**
"Aspect" Only	**13.70%**	483.7%	264.4%
No Aspect	12.71%	430.3%	216.1%

Table 2

Avg Absolute Error for Different Output Types and Selections of Features			
	Avg Ratings	# of Reviews	Avg # of Reviews per Day
All Features	**0.3611**	79.476	0.0965
Numeric Only	0.3800	75.907	0.0941
Text Only	0.3768	96.264	0.1101
"Aspect" Only	0.4065	92.687	0.1092
No Aspect	0.3637	80.694	0.0956
All Features	0.3790	76.337	0.0974
Numeric Only	0.3766	**71.282**	**0.0825**
Text Only	0.3903	**101.115**	**0.1219**
"Aspect" Only	**0.4102**	95.848	0.1157
No Aspect	0.3799	75.391	0.1012

Table 3

Overall Accuracy

We first noticed how well our model performed at predicting the average ratings of future reviews. All error types in all cases are quite low, and, according to the average absolute error, in all cases is less than half a "star" out of five. Of particular note is how well using only the textual features performed. This would imply that only the text someone writes in a review may be needed to numerically rate an establishment. This is useful since comparing numerical ratings is easier than comparing text reviews, but an establishment may only have access to text reviews.

We also noticed how poorly our model performed at predicting the number of future reviews. This is most likely due to the relatively small number of reviews for most hotels in our data set. However, the number of reviews is not a very robust metric in the first place since it is considering the number over a theoretically infinite period of time after the cutoff date (if we were to consider real time data). This is not very realistic since a hotel will never stay in business for an eternity, and each hotel will stay in business for a different amount of time.

A more robust metric to consider would be the average number of reviews per set period of time after the current date. This handles the infinite time and varying times different hotels stay in business. In particular we considered number of reviews per day. In regards to this metric, our model performed much better, but still showed a high degree of percentage error. Again this is most likely due to the relatively small number of reviews for most hotels in our data set.

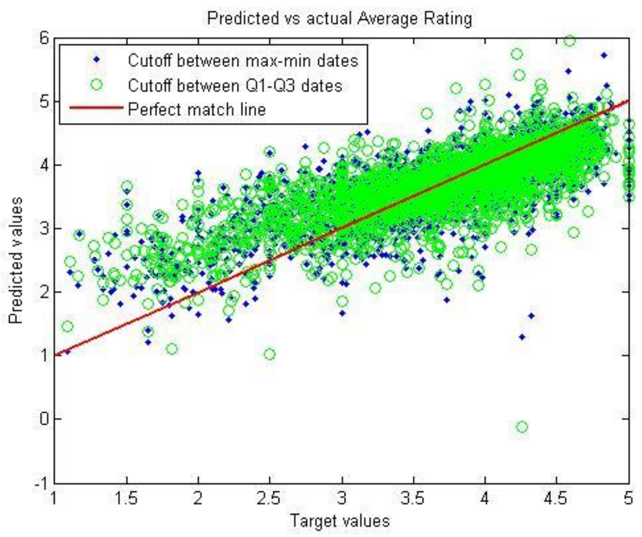

Figure 3

Comparing data subsets

Generally speaking when looking at the error values across all tables, error values for set 1 are lower than the corresponding error values for set 2. This shows that having more data is better for accurate regression even if it has more variance, as long as the variance is not significantly higher.

Comparing feature subsets

As expected, using more features tends to produce more accurate results, especially when predicting the average ratings of future reviews.

Another conclusion we drew was that with respect to the average ratings of future reviews, the textual features alone are better at prediction than the numeric features alone. This means that the text is providing a larger amount of information than the numeric features are about how a reviewer feels about a particular hotel.

Conversely, the numerical features perform better than the textual features at predicting both the number of future reviews and average number of future reviews per day. This makes sense since the numerical features include information about the number of previous reviews and the time span over which those reviews were obtained while the textual features should be less correlated with the number and time of reviews.

Finally we determined that the textual features from the aspects perform worst of all by themselves. This implies that there is less useful information in those features than the rest of the textual features. This makes sense since the textual features from the aspects are mostly nouns which do not usually say much by themselves about the value or worth of a particular establishment.

Conversely, the feature subset containing the textual features from NLTK analysis and the numerical features performs almost as well as using all the features. This implies that using an intelligently reduced feature set can allow for only slightly reduced accuracy, but also reduced learning and predicting time. This is especially useful if using a more complicated basis function or more data.

Considerations for Future Work

With more time to analyze the text of the reviews, it should be possible to extract opinion-related features from the text, similar to the sentiment analysis performed in Hood et al [1]. With the above, and access to better/more data (i.e. the data itself has more numerical features), it should be possible to create a feature subset that contains more information than what is in the features we used, which would allow for better prediction. Furthermore, additional methods for finding such a feature subset (e.g. PCA) could be considered.

Additionally, with more learning time it should be possible to consider more/more varied basis function types (e.g. higher degree polynomials) and regularization values.

Finally, selecting a cutoff date using a continuous feed of real time data (i.e. the TripAdvisor site itself) would give us more reviews per hotel (even with the assumption that reviews are not continuously created), thus reducing data loss.

References

[1] B. Hood, V. Hwang and J. Kang, "YelpDatasetChallengeWinner_InferringFuture.pdf," 2013. [Online]. Available: http://www.yelp.com/html/pdf/YelpDatasetChallengeWinner_InferringFuture.pdf. [Accessed 1 December 2014].

[2] W. Hongning, L. Yue and Z. ChengXiang, "Latent aspect rating analysis without aspect keyword supervision," 2011.

[3] F. Qianqian, "JSONlab: a toolbox to encode/decode JSON files in MATLAB/Octave - File Exchange - MATLAB Central," [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/33381-jsonlab--a-toolbox-to-encode-decode-json-files-in-matlab-octave. [Accessed 22 11 2014].