PREDICT TRIPADVISOR HOTEL RATINGS BASED ON REVIEWS

Author: Angie Dinh

Introduction

The goal of that project is to predict rating of a hotel on TripAdvisor based on their reviews.

Data collection

The data is collected by scraping reviews and ratings from TripAdvisor webpage. In this particular project, I only collect data from New York city. The source page can be found here: https://www.tripadvisor.com/Hotels-g60763-New_York_City_New_York-Hotels.html

I only collect a small sample for academic purpose and got permission from TripAdvisor. To ensure that the data is unbiased even though I only collect a sample, I write my code to go to random review pages for each hotel.

I use Python for web-scraping and initial data pre-processing. As for web-scraping, I use the BeautifulSoup library with a small implementation of multi-threading for faster code. Scrapy can be a more efficient tool, but I use BeautifulSoup for the purpose of learning web-scraping. The multi-threading implementation is not fully correct, but it serves the purpose of getting a small sample of the data.

In the code, I run loops through hotel and review pages because they follow this pattern:

For each hotel:

The first review page:

Hotel_Review-g60763-d223024-Reviews-Hotel_Chandler-New_York_City_New_York.html#REVIEWS

The second review page:

Hotel_Review-g60763-d223024-Reviews-or10-Hotel_Chandler-New_York_City_New_York.html#REVIEWS

The next page is [text]-or20-[text], and the following pages follow similar patterns.

For code, please refer to the following files: TripAdvisorWebScrape.py and TripAdvisorDataCleaning.py.

Data cleaning and text mining

After the initial data pre-processing, I now have a .csv file that contains ratings, reviews, and review headers for each review. Review header is the header before each review, as one can see from the page.

I use the “tm” package in R to convert the reviews and review headers into two Corpuses, from which I perform string processing (convert to lower case, remove punctuation, remove spare terms,… etc) and create two word frequency tables.

The word frequency tables are formatted like the following example:

Review/Word	Rating	bad	good	great
Review 1	5	0	3	3
Review 2	3	1	1	0

In the real dataset, the words are ordered alphabetically. However, the idea is similar. Each column is a word, each row is a review, and the cell values are the frequencies of each word (column) in each review (row).

Then I merge the two tables and weigh the review header frequency three times the review frequency, because I believe the review header summarizes more of the reviewer’s opinion about the hotel.

For the text mining code, please refer to this file: TripAdvisorTextMining.R

Model

I use Naïve Bayes as the model, for the following reasons:

- This is a classification problem; therefore, a linear regression is not appropriate
- There should be no clear linear relationship between frequencies of so many words in a review and its rating, thus the problem is not linearly separable and the logistic regression is not appropriate
- The number of features is too large (compared to the number of observations) to fit a decision tree.

- This leaves us with Naïve Bayes, nearest neighbor, and support vector machine
- Since the training set is small (less than 30,000 observations) and the number of features is large (over 6000 variables), a high-bias, low-variance model like Naïve Bayes is usually a better choice.
- I try the support vector machine and it runs very slowly. Going on forward with this project, I would want to try fitting an SVM using a more efficient way.
- In the literature, Naïve Bayes has always been a good method for text classification

1. Naïve Bayes algorithm:

In the training set:

For each review X:

Find the conditional probability that each word appears in each class: $P(X_j = a_{jz}|c_i)$

X_j: attribute, in the case, users'reviews

a_{jz}: distinct values of attribute X, in this case, word frequency

c_i: class, in this case, users' ratings from 1 to 5

end for

In the testing set:

For each observation X:

Calculate:

$$P = P(X_1 = a_1|c_i) * P(X_2 = a_2|c_i) * P(X_3 = a_3|c_i) * \ldots .* P(X_n = a_n|c_i) * P(c_i)$$

Then find class i that gives the highest value of P

End for

2. Modification of the algorithm:

There are problems with the built-in Naïve Bayes algorithm; therefore, I decide to implement another version of the Naïve Bayes algorithm, using the formulas described in the paper by Karger, Rennie, Shih, and Teevan[1].

a) Find the conditional probability that each word appears in each class:

I use the following formula, which is more suitable for natural language processing:

In each class:

For each word:

$$P(X_j = a_{jz} | c_i) = \frac{Frequency\ of\ that\ word\ in\ the\ class +\ 1}{Frequencies\ of\ all\ the\ words\ in\ the\ class + Vocabulary\ Size} \quad (1)$$

Vocabulary size is the total number of different words in the training set, which is the number of training set columns, in this case.

Vocabulary size is the total number of different words in the training set, which is the number of training set columns, in our case.

b) Using log probability:

When I start predicting, the model involves a lot of multiplication of small probabilities, which brings the result to 0. I fix this problem by using the log of probability, which turns the multiplication into the summation. We can do that because one can prove that if p1 > p2, then log(p1) > log(p2) and vice-versa. Therefore, the class that gives the maximum of log probability also gives the highest probability.

c) Using one-versus-all Naïve Bayes (OVA) to solve the skewed data bias:

I realize that in our data, the majority of the observations are in class 5 and few of them are in class 2. The log probability is negative, the log probability is set to be 0 when a vector does not appear in a given class, and there are two few class 2; therefore, class 2 always have the highest probability for all the testing observations, thus the prediction is incorrect.

To solve this problem, I use one-versus-all Naïve Bayes. The formula is as follow:

[1] Karger, Rennie, Shih, Teevan: http://people.csail.mit.edu/jrennie/papers/icml03-nb.pdf

$log_{OVA}(P(x\ in\ c_i)$=

$$\left[\log(p(c_i)) + \left(\sum \log(conditional\ probability\ \ (instance\ x = x(1,2,3\ \dots)|class\ i)) - \sum \log(complement\ conditional\ probability\ (instance\ x = x(1,2,3,\dots)|class\ i)\right)\right]$$

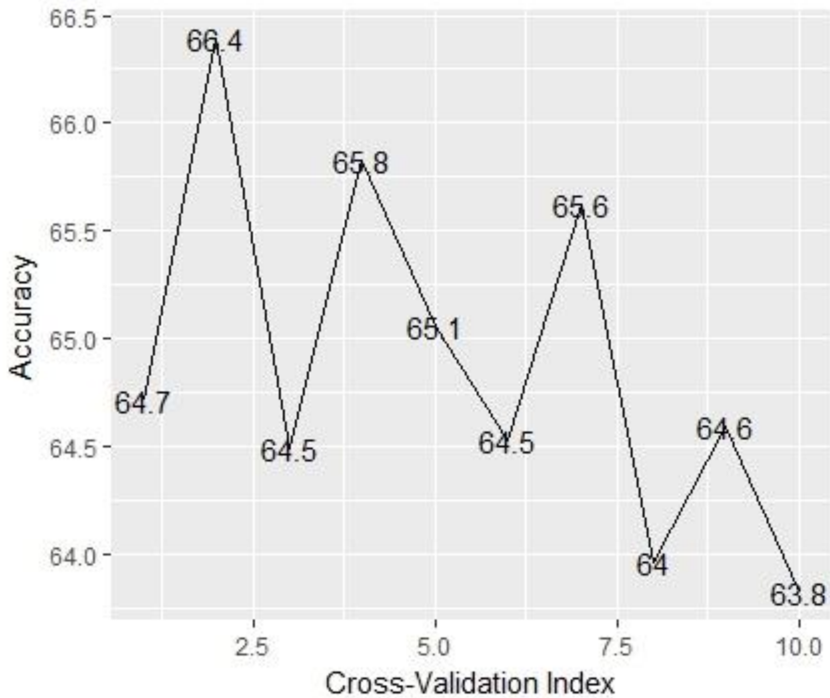
Conditional probability (instance X | class i) are the values that I calculate in part a. Complement conditional probability is calculated as follow:

$$Pcom(X_j = a_{jz}|c_i) = \frac{Frequency\ of\ that\ word\ in\ classes\ other\ than\ c(i) +\ 1}{Frequencies\ of\ all\ the\ words\ in\ classes\ other\ than\ c(i) + Vocabulary\ Size} \quad (3)$$

Results

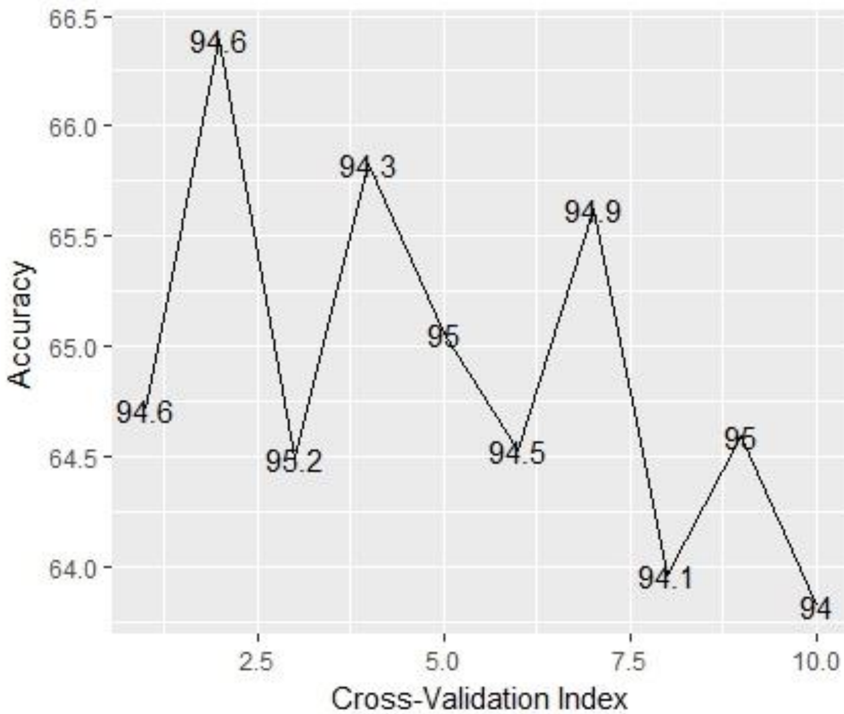
Cross-validation test accuracy:

If we maintain the strict classification accuracy criteria, which means that if the true rating is 4, the predicted rating has to be 4 to be correct, the cross-validation test accuracy are shown below:



The accuracy is around 65%. The base model, which is predicting every rating to be 5, has a 49% accuracy. This model is better than the base model

The performance of this model is better when if I make the classification accuracy criteria a little less strict. In this case, if the predicted rating is one value above or below the actual rating, that prediction is still considered correct. In this case, the cross-validation accuracy is shown below:



The accuracy rate around 95%, which is very high compared to the 79% accuracy of the based model (predicting all ratings to be 5).

For code, please refer to the file TripAdvisorProject-NaiveBayesModel.R

Ways for improvement

There are still many ways to improve this model to be improved. Going forward with this topic, I would like to explore semantic meaning of the words further and see if there are any features (frequencies of particular words) that can be eliminated. I would also like to fit other models and compare their performances, and to find a way to efficiently train a support vector machine model.

References

Karger, Rennie, Shih, Teevan. "Tackling the Poor Assumptions of Naive Bayes Text Classifiers." *Proceedings of the Twentieth International Conference on Machine Learning* (ICML). 2003.