Chinese restaurant text analysis

1. Introduction

Our group mainly focus on the reviews of chinese restaurants in U.S. The goal of our analysis is to get the information of each restaurant in three aspects: food, service, and price.

The method we use to achieve our goal is to score the adjectives in all reviews, and extract the adjectives describing these three specific words: 'food', 'service', and 'price'. And we use the average score of these adjectives as the score of the restaurant in this aspect.

We have aslo introduced the Factor of Credibility to help us get more reasonable average scores, and used correlation coefficients to help us bulid a more informative weighted rank table. The final rank table tell the restaurant owner the rank it got in each aspect, and an overall rank considering the importance of each aspect as well.

2. Data Description & Pre-analysis

In total there're 209897 reviews for 2954 chinese restaurants located in 13 states, and aroud 60% of them are 4 or 5 stars.

We firstly performed some pre-analysis on the given business information of these chinese restaurants. Some attributes have been transformed into a more analysis-friendly format. e.g. we have transformed the attribute 'Business Parking' to 3 parking-friendly level (1: not friendly, 2: friendly, 3: super-friendly).

Based on EDA we got some general findings, e.g. parking-friendly restaurants or restaurants providing free WiFi would get more stars in average. But besides these overall findings that may also apply to other kinds of restaurants, we got some interesting findings which may reflet the characteristic of chinese restaurants in china.

We can find that, compared with other kinds of restaurants, most chinese restaurants don't provide TV, free WiFi and outdoor seats, the proportion of $3\ or\ 4$ restaurants is also smaller. All of these findings indicates that chinese restaurants in the U.S seems to be a place only to fill your stomoach, quickly and coveniently. Based on this finding we did some simple word frequency regression on reviews of the 'famous' restaurants QQ Express, as well as the reviews of restaurants in State WI & OH.

3. Thesis Statement

The regression result from pre-analysis (top positive/negative words) lead us to our final analysis goal: Getting the information of each restaurant in these three aspects: food, service, and price.

The method we use to score each restaurant in these three aspects is to extract the scored adjectives describing these three words: 'food', 'service' and 'price'.

4. Scoring for Food, Service, and Price

4.1 Extract all adjectives in reviews

The first step is to extract all the adjectives appeared in all reviews.

To prepare for the adj-extraction, we've done some preliminary process over the reviews, some of them are shown as below (Because we will tag words in this step, thus punctuations and stopwords need to be kept for the integrity of a whole sentence):

$$not\ great \rightarrow notgreat$$

$$high\ quality \rightarrow great$$

$$5\ stars \rightarrow fivestars$$

Then we use the function pos_tag() from nltk package in Python to help us tagging the words and extracting adjectives from them, here is an example:

$$\text{It's a fivestars restaurant!}$$

$$\rightarrow [\text{'It','PRP'}], [\text{" s ","VBZ"}], [\text{'a','DT'}], [\text{'fivestars','JJ'}], [\text{'restaurant','NN'}], [\text{'!','.'}]$$

$$\rightarrow \text{'fivestars'}$$

We only kept words appeared more than 100 times, and finally we got an adjective list with 1851 words.

4.2 Get the sentiment score of each adjective

In this step we use a logistic regression based on word(adjtives actually)-frequency to score each adjective.

First we removed all the punctuations and stopwords, and delete words like 'pretty', 'very', and 'always' (most of them are describing an adjective but could be regarded as adj by nltk).

Then we performed a word-frequency logistic regression, where

$$Y = \begin{cases} 1, & star \geq 4 \\ 0, & star < 4 \end{cases},$$

X = frequency sparse matix only with words in adjList

We didn't put any penalty on the regression coefficients to prevent over-fitting problem, because our goal here is to get the sentiment score for each adjective rather than precise prediction.

We only keep words with the absolute value of coefficient $\geq$ 0.2, and delete some weird words such as 'bit' and 'often'. Then we use the regression coefficients as the sentiment score for each adjective.

Finally there're 975 words remaining, with 462 positive words and 513 negative words.

4.3 Break the sentence

To make the result of finding describing adjectives as precise as possible, we not only set punctions (e.g. ,.?!) as sentence breaker, but also set specific words like 'but', 'otherwise' to break sentence:

Price is reasonable but service is notgood, the food tastes okay.

$\Downarrow$

Price is reasonable || service is notgood || the food tastes okay.

4.4 Score each restaurant

With the scored adjList and sentence breaker, we can now extract **the scored adjective** near specific words for each restaurant. Take our favorite QQ Express for example, how is the word **'price'** being described?:

```
        it      gives        you          a      wider
 0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 selection         of      items         at        the
 0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
comparable      price
 0.5797432  0.0000000
```

Calculate the average score of the adjectives near **'price'**, then we get the score of price for QQ Express:

```
> score_summary(score_result[[1]], score_result[[2]])[[1]]
reasonable      worth       nice      solid       good
 0.8307113  0.6871596  0.4174705  1.2248443  0.6658830
comparable       good    average    average  excellent
 0.5797432  0.6658830  0.0000000  0.0000000  1.3567568
> score_summary(score_result[[1]], score_result[[2]])[[2]]
[1] 0.6428452
```

Repeating the process shown above, we calculated the average scores near the words 'food', 'service' and 'price' for each restaurant.

4.5 Factor of Credibility

It's likely that a restaurant with rather averge service would get really high score in 'service' because only one or two people have mentioned service in their reviews, coincidentally they thought the service is great. Of course the high score in this situation is not valuable at all. To prevent this from happening, we introduced the **Factor of Credibility** to indicate how credible the score is:

$$C_{price,res} = \frac{\#\,'price'}{\#\,reviews_{res}}$$

The **Factor of Credibility** was the proporion of reviews contained one specific word among all reviews for a restaurant. And we used it to help us recalculate the average scores:

$$Price_{res,new} = Price_{res,old} * C_{price,res}$$

4.6 Which aspect is the most important?

We only kept restaurants with more than 20 reviews, and recalculated the new average score for each of them, then we can get the Madison's top 5 restaurants in each aspect:

Price

Rank	Restaurant
1	QQ Express
2	Hong Kong Wok
3	Jade Garden
4	Happy Wok
5	Double 10 Mini Hot Pot

Service

Rank	Restaurant
1	Double 10 Mini Hot Pot
2	Tavernakaya
3	Umami Ramen & Dumpling Bar
4	Imperial Garden Chinese Restaurant
5	Great Wall

Food

Rank	Restaurant
1	ZenZen Taste
2	Chili King
3	Hong Kong Wok
4	Double 10 Mini Hot Pot
5	Double 10

However, which word should the business owner value most? To answer this question, we ranked all the restaurants in the U.S, and calculate the average stars it got from customers. Then we calculate the correlation coefficient between the rank in one word with the average stars it get, which indicates the importance of this aspect. The result is shown as below:

$$\rho_{food} = 0.69, \rho_{service} = 0.57, \rho_{price} = 0.17$$

4.7 Get the final, modified, weighted rank table

With the correlation coefficients we got, the modified rank considering the importance of each aspect should be like:

$$Rank_{price,new} = Rank_{price,old} * \rho_{price}$$

After recalculation we can get the modified rank table. Summing the modified rank for each word, the retaurant with smallest rank-sum should be the best.

Then we resorted the original rank table by the modified rank-sum result, take a brief look at the final result:

	food	service	price	final_rank
Double 10 Mini Hot Pot	4	1	5	1
Chili King	2	6	32	2
Double 10	5	11	14	3

5. Conclusion & Strong/Weak Points

5.1 Conclusion

For chinese restaurants, taste, service and price are all important. But it seems that taste and service weighs more than price in getting more stars from customers.

For each restaurant, it can get its rank in these three aspects as well as the final overall rank, indicating the business owner which aspect should be improved.

5.2 Strong Points

We have introduced the **Factor of Credibility** when calculating the average score for each words, which prevents the unreasonably high/low scores caused by only few reviews.

We calculate the correlation coefficients between the rank of the word and the stars, and use them to produce the final weighted rank table, which gives business owners an informative rank considering the importance of different aspects.

The business owner could not only get the rank of his/her restaurant, but also get more detailed information from the adjs/sentences about why the restaurant gets high/low rank in this aspects.

5.3 Weak Points

There do exist many other ways to describe the taste, service and price, which would be missed by the adj-extacting method.

There should be many other aspects which may affect the stars got, such as dim sum for Nani Restaurant, which were not included in our analysis .

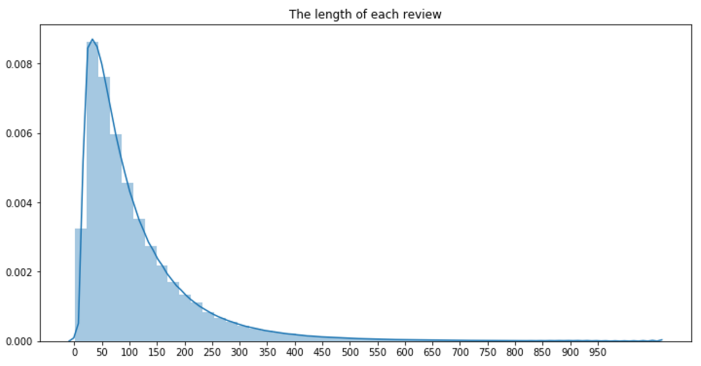
Of course, the final scores we get from the adj-scoreList and sentence breakers are informative, but not the most accurate.

6. Prediction

6.1 Motivation

Our goal is to predict the rating of the reviews mainly based on the text data. Since it belongs to parts of NLP problem, we choose the most popular model called LSTM to solve this sentiment classification problem. In general, LSTM is an improved version of RNN, which is used to solve the sequential problem and at the same time overcome the long-term dependencies defect.

6.2 EDA



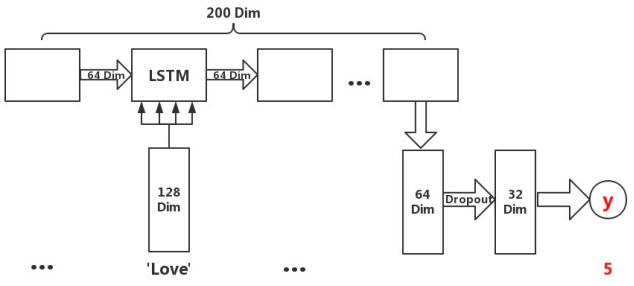
From above, we can find most reviews are less than 200 words.

6.3 Tokenize

As we all know, the computer can't recognize the whole words. Thus, we transform the text into numbers which represent each word through the Tokenize method. Since the average adult knows 20000 to 35000 words, we choose 20000 top frequency words.

6.4 LSTM

Our model constructed via four main layers. The Embedding layer is used to vectorize the chosen words in our corpus, and it overcomes the high-dimension sparse problem caused by the one-hot encoder. The LSTM layer is used to solve the sequence problem and solve the vanishing gradient problem by the gate functions. The Dense layer aims to make the model more robust based on feature fusion. The Dropout layer can be applied to handle the overfitting problem.



6.5 Application

In our model, the first layer is the Input layer, which has 200 dimensions, that is to say, all reviews have a length of 200. Next, it's Embedding layer, which transforms the words into vectors and the size is 128 dimension, and the total number of parameters are $20000 * 128 = 2560000$(because of the number of top frequency words is 20000). Next, it's the LSTM layer, and the output size is 64 dimension. Since the LSTM cell has four gates, which means the total number of parameters are $(128 + 64 + 1) * 4 * 64 = 49408$. Next, it's Dense layer, that is to say, the fully connected layer and the total number of parameters are $(64 + 1) * 32 = 2080$. Next, it's Droupout layer, which used to avoid overfitting problem. The last layer is a Dense layer, here the activation functions are 'linear' and 'softmax' respectively for different loss functions, and the total number of parameters are $(32 + 1) * 1 = 33$ and $(32 + 1) * 5 = 165$ respectively.

Layer	Output Shape	Param #
Input	(None, 200)	0
Embedding	(None, 200, 128)	2560000
LSTM	(None, 64)	49408
Dense	(None, 32)	2080
Dropout	(None, 32)	0
Dense(linear)	(None, 1)	33

Layer	Output Shape	Param #
Input	(None, 200)	0
Embedding	(None, 200, 128)	2560000
LSTM	(None, 64)	49408
Dense	(None, 32)	2080
Dropout	(None, 32)	0
Dense(softmax)	(None, 5)	165

Method	Loss	Accuracy	Score On Kaggle
MSE	0.6525	0.5332	0.61528
CrossEntropy	0.7698	0.6810	0.69130

From the first two tables above, we tried two different loss functions, the MSE, and CrossEntropy respectively based on the same neural network frame. The last table is our final result based on the model we trained.

Contribution:

Luwei Liang: prediction, notebook, model building, coding

Yilun Zhang: prediction, notebook, summary, coding

Ruyi Yan: prediction, notebook, pre-analysis, coding

Reference1: https://www.kaggle.com/sbongo/for-beginners-tackling-toxic-using-keras (https://www.kaggle.com/sbongo/for-beginners-tackling-toxic-using-keras);

Reference2:http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)