Suggestions for merchants and rate prediction based on Yelp dataset

1. Introduction

1.1 Motivation and thesis statement

Yelp is a popular software in people's daily life. For merchants, yelp reviews play an important role in business operations. We can obtain information about user experience and sentiment by analyzing the text of yelp reviews. So the goal of this project is to provide useful, analytical insights to business owners on Yelp and, based on these insights, make actionable suggestions to owners in order to improve their ratings in Yelp. We also propose a prediction model to predict the ratings of reviews based on the text and related attributes.

1.2 Background information about data

For review data, the training dataset is based on 5,364,626 reviews. It contains business_id, date, stars and text. In the test and validation data, there are 1,321,274 elements without rating. For the business data, there are 154,606 businesses in the training dataset, which contains specific information about businesses' attributes, business_id, categories, city, hours, is_open, latitude, longitude, name, postal_code, state, and 38,000 observations in the test dataset.

2. Data pre-processing

2.1 Data cleaning for the text of review data

(1) Convert all uppercase letters to lowercase.
(2) Replace all commas with periods, since we use a period to seperate a relatively emotionally independent statement in the following analysis.
(3) Convert words with negative meaning, e.g., n'/n't $\rightarrow$ not.
(4) Keep special symbols including ! and combination of !, ., ..., !?, :).
(5) Delete stopwords based on nltk[1] stopword corpus except: i) words with negative meaning: not, no, nor; ii) words that represents a third person: he, him, his, himself, she, her, she's, her, hers, herself, they, them, their, theirs, themselves.
(6) Lemmatization: use a dictionary to return the original word of verb and noun, e.g., ran $\rightarrow$ run, timing $\rightarrow$ time.
(7) Negation: add _NEG to each word between the negation and the following first period.

2.2 Dataset selection

We filter the business_id of the "categories" that contains "Brunch" in the business data and correspond it to whole review data. Then we obtained the brunch review data which includes 521673 observations of 4322 businesses.

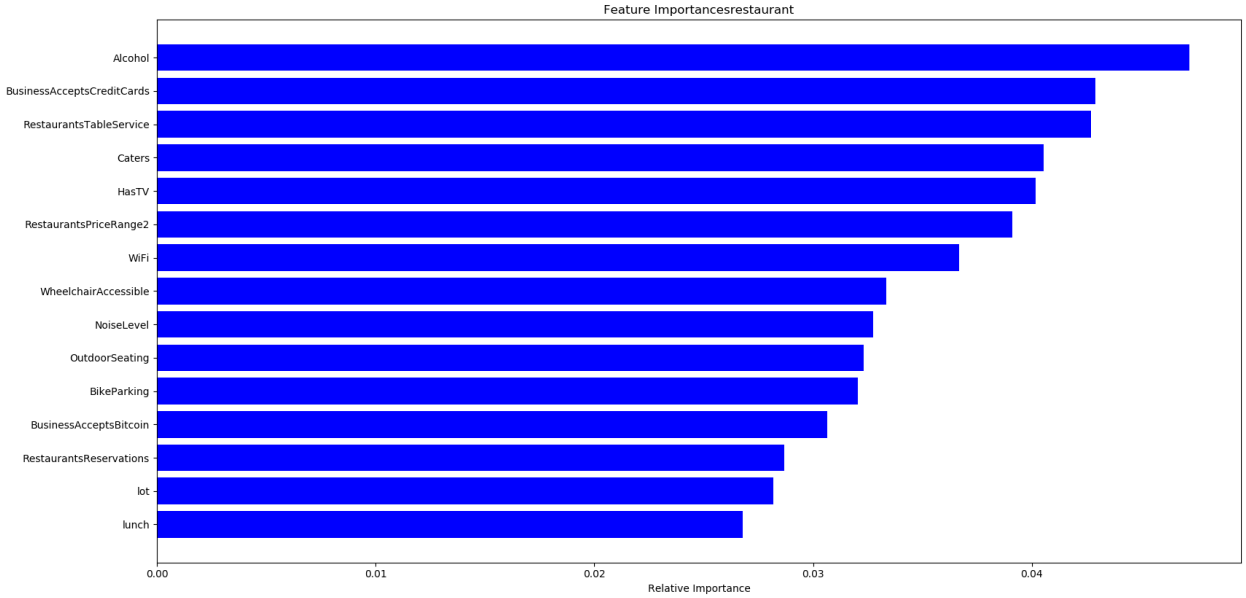
3. Goal 1: Suggestions for Brunch Restaurants

3.1 Based on Attributes

In the Brunch business dataset, each observation of "attributes" contains many different factors. And for each factor, it contains different levels. For example, for the factor "GoodForMeal", it contains 6 options, 'dessert', 'latenight', 'lunch', 'dinner', 'brunch', 'breakfast'. For each option, it also contains different levels, e.g., {'breakfast': False; 'breakfast': True}. We converted all the options of attributes into independent columns and get 72 different attributes in total. Then we use Random Forest method to obtain feature importance of attributes.We considered two sets of data. One contains all the business attributes related to restaurants, the other one contains all the reviews related to brunch restaurants.We'll try to find the most important feature for brunch restaurants and restaurants respectively.

3.1.1 Random Forest-Feature Importance

We fit a random forest model on the data and calculate the feature importance for each attributes on both datasets based on mean decrease impurity, as the impurity can be treated like a meassure of the information one attributes contains. And then we compare the top 15 attributes with high feature importance on both datasets. We can see these two lists of attributes are quite similar.



3.1.2 Linear regression

For different levels of top 15 attributes with high feature importance, we build a linear regression with levels as independent variables and different star ratings as dependent variables. For example, the attribute "WiFi" contains 3 levels, 'paid', 'free' and 'no'. Its linear regression coefficients are as follows. Compared with feature 'free', the coefficients of 'paid' and 'no' are both negative, so we could think it's better for a brunch restaurant to have free WiFi. And as option 'paid' has samller coefficient, it's the worst option for WiFi.

variable	(Intercept)	no	paid

variable	(Intercept)	no	paid
Estimate	3.874704	-0.115175	-0.25036

3.2 Based on Keywords and Reviews

3.2.1 Latent Dirichlet Allocation (LDA) Model

LDA model treats documents as a mixtures of topics that constructed by words with certain probabilities. It assume the document is written in the following order. First, decide the number of words in the topic. Second, choose a mixture of documents with different topics. Finally, given a topic, and choose words based on a conditional distribution.

It's an unsupervised learning model, very similar to clustering, and we would use topic coherence to evaluate the model. The key parameter here is the number of topics. We train the model on 3 topics. This value is selected by comparing the topic coherence value. However, the readablity is still not good enough. We need to analysis the outcome together with the bigram and trigram methods.

3.2.2 Bigram and Trigram

As we delete most of modal word and conjunction as stopping words, a bigram and trigram model is possible to combine the adjectives and nouns directly. This will help to reveal the direct sentiment towards our key word.

3.2.3 Hypothesis test

We find the high frequency words based on CountVectorizer and respectively select the top 20 words related to food, drink, atmosphere and them build the wordlist. In the brunch review data, for each keyword, we filter out the review texts and star ratings that contain it. We divide the reviews into two types: 1. the reviews contain the keyword; 2. the whole brunch review. We use these two type of data to conduct hypothesis test:

We use Chi-square test to test if the distribution of star ratings in these two type of data have significant difference. Since the we derived the star ratings corresponding to each word is significantly different from the star ratings of the whole brunch review.

The results of keyword "cocktail" and "benedict" are as follows:

-	cocktail	benedict
Chi-square statistic	574.08	2030.6
P-value	$6.29e^{-123}$	0

We also use t-test to test if the average star ratings in these two type of data have significant difference and divide the results into no difference and with differences. In the type of "with difference". We use one-side t-test to proof whether the average star rating is higher or lower than the original star rating. If the average star rating is

4. Goal 2: Prediction

We use all the train reviews to build a multinomial logistics regression with TF-IDF model with 5 results (1-5 star), and the final RMSE of the test set is 0.81245.

4.1 Bag of words

The bag of words model assumes that we do not consider the context between words and words in the texts, only the weights of all words. The weight is related to frequency of each word appears in the texst.

First, we tokenize the text into words. After tokenization, by counting the frequency each word appears in the text and extracting feature words by Chi-square test, we can get the word-based feature of the texts. Placing the words of each text together with the corresponding word frequency is the vectorization. After the vectorization, we use TF-IDF to perform weight correction of the features, and then normalize TF-IDF. After performing these feature extraction, we apply the training data to the machine learning model.

4.2 CountVectorizer/ TF-IDF

CountVectorizer[2] converts the words in the texts into a word frequency matrix. For example, the matrix contains an element $a[i,j]$, which represents the word frequency of the *j word under the i-th* text.

TF-IDF tends to filter out common words and retain important words.

Let w, t denote a word and a text, TF-IDF (Term Frequency $\times$ Inverse Document Frequency) is calculated by

$$\mathrm{TF}(w,t)=\frac{\#w \text{ in } t}{\#\text{words in } t},\quad \mathrm{IDF}(w,t)=\log\frac{\#\text{words in } t}{\#\text{texts that contain } w},\quad \text{TF-IDF}=\mathrm{TF}\times\mathrm{IDF}$$

Then we apply normalization to the TF-IDF algorithm. Since high TF-IDF of some words may correspond to higher similarity between texts, normalization helps us interpret the similarity score better from the tf-idf score, it is also useful to utilize the tf-idf score as a feature in our classifier model.

4.3 Variable selection

In our project, at first we screen out high frequency words in test and validation data by CountVectorizer, since we think the low frequency words, even if they are important, will only work for very small amounts of data, not make a big difference to the whole. Next, we correspond the high frequency words selected from the test and validation data to the training data, and use them to conduct variable selection in the training data based on Chi-square test. Based on the value of chi-square statistic, we choose the top 2000 significant words of training data to build wordlist. Similarly, we also use the feature importance score of random forest, and select 2000 significant words.

4.4 Model fitting

For the cleaned training data, we apply the sparse matrix whose columns represent words values TFIDF to logistics regression.

We use multinomial Logistic regression with optional L2 regularization in scikit-learn.[3] For the model parameters, we choose 'multinomial' over 'one vs rest' for better accuracy, normalize the data and use Stochastic Average Gradient solver for faster convergence speed and better robustness. As for regularization strength C, we use prediction accuracy with grid search and 5 fold cross validation to set it to 1.

4.5 Model evaluation

We use the learning curve to verify the robustness of the model. From the learning curve plot, we could see the score is approximately converges to 0.682, which proves the low bias property. With the increase of training sample set, the train score decreases and cross-validation score increases and come closer, which proves the low variance. These indicates the Logistic regression model is robust and doesn't suffer from overfitting or underfitting.

5.Conclusion

5.1 Suggestions

For the business attributes, Most of the results are quite intuitive: Beer and wine is the best option for alcohol, even better than a full bar. The restaurants which are able to cater tend to have better reviews. The restaurants have TVs, free Wifi, out door seating options and wheel chair accessiblilty tend to have better reviews. And expensive restaurants and quite restaurants tend to have better reviews.

Serveral results need some assumptions to understand: The restaurants which don't accept credit cards tend to have better reviews. We suppose it's because these are very small restaurants run by individuals. They need to provide really good food to attract people.

A brunch restaurant that supplies cocktail usually have a higher rate. Multiple selections of alcohol and a cocktail menu usually means a higher rate. If appetizer is available, shrimp cocktail is a good selection. Special cocktail sauce and signature cocktail are essential conditions for a good brunch in a cocktail business.

5.2 Strength and weakness

Strengths: 1. We use both business attributes and keywords in the reviews to make suggestion, it is feasible and sufficient. 2. Through tuning parameters, our prediction model has been improved in robustness.

Weakness: 1. We need to interpret the code results manually, which is not completely automatic. 2. We should fit more models for rating prediction and compare them.

5.3 Contributions

- **Hongyi Jin**: Equally
- **Jingyu Ji**: Equally
- **Jiaming Zhou**:Equally

5.4 References

[1] Nltk. Retrieved from https://www.geeksforgeeks.org/removing-stop-words-nltk-python/ (https://www.geeksforgeeks.org/removing-stop-words-nltk-python/)
[2] CountVectorizer. Retrieved from https://scikit-learn.org/stable/modules/feature_extraction.html (https://scikit-learn.org/stable/modules/feature_extraction.html)
[3] Scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)