Satisfaction Classification on Hotel Reviews

Du Xiang
University of California, San Diego
dxiang@ucsd.edu

Jinghong Luo
University of California, San Diego
jil040@ucsd.edu

Yifei Wang
University of California, San Diego
yiw014@ucsd.edu

ABSTRACT

This paper analyses a dataset of European hotel reviews from Booking.com and tries to determine the user's experience of the hotel, such as satisfied, neutral, and dissatisfied given the fields of the data point (review). It attempts to perform the task by building a Logistic Regression model and comparing its accuracy to the baseline and other models with different sets of features to find the best one for user satisfaction level classification.

1. INTRODUCTION

As transportation methods are rapidly developing, the demand for travelling is increasingly high nowadays. Hotel industries are also growing along with the trend, whether it's for business or leisure. Meanwhile, the problem of picking the right hotel could be somewhat challenging. The ratings of the hotel can be somewhat untrustworthy given that some platforms are paid to change ratings in favor of the hotels. Therefore, it's always better to directly read the user reviews.

In this study, we will try to build a machine learning model to help process the text and reviews to assess the user experience (rating). Since the user reviews are broken down into numbers from 0 to 10. It's hard for naive hostelbookers to know if a score of 7 is considered as good or bad. So, instead, we will map their scores into categories based on the seen distribution. In this way, we can directly predict the user's satisfaction (satisfied, neutral, and dissatisfied) from the review texts.

The task/study is splitted into five parts. In the first part, we will be examining the dataset and its respective fields. Next, we will perform exploratory analysis on the dataset and try to find the feature relationships. In the third part, we will be describing our predictive task and the process of feature extraction. Part 4 will show the variations of our model and compare the results of different approaches. Lastly, we will be making the conclusion and summarizing our findings related to the other literatures.

2. DATASET

2.1 Identify a dataset to study

We use a dataset from Kaggle that contains a total of **515,738 reviews** from users to the hotels in Europe on Booking.com to perform this study. We choose this dataset mainly because this dataset is well labeled, and we are curious how the user experience changes based upon the different elements in the user review.

Due to memory issues and limited computational power, we decided to randomly select 40% of the whole dataset (204,988 reviews) as our primary dataset. We then separate the primary dataset to use 180,000 reviews as our training set, and 24,988 for our validation set. For the test set, we will randomly select 20,000 reviews from the unused dataset.

2.2 Exploratory Analysis

First, we will check if there is any missing data in our dataset. We found that there is very little dataset that's missing in the 'lat' and 'lng.' Since the location data is missing at random, and the amount of missing data is less than 1%, we decided to drop them.

2.2.1. Summary Statistics

Briefly looking at the dataset, we find that each data point contains the following features.

Field	Description
Hotel_Address	The address of the hotel in Europe
Additional_Number_of_Scoring	The number of valid scores without review
Review_Date	The date user gives this review
Average_Score	The average score of the hotel
Hotel_Name	The name of the hotel
Reviewer_Nationality	The nationality of the reviewer
Negative_Review	The negative review user gives
Review_Total_Negative_Word_Count	The number of words in all negative reviews
Positive_Review	The positive review the user gives
Total_Number_of_Reviews	The number of words in all positive reviews
Review_Total_Positive_Word_Count	The number of words in all positive reviews
Reviewer_Score (out of 10)	The overall scores the user gives
Tags	The word tags user gives to the hotel
lat	The latitude of the hotel's location
lng	The longitude of the hotel's location

With the help of the folium package, we are able to generate the distribution of the hotels in the map.

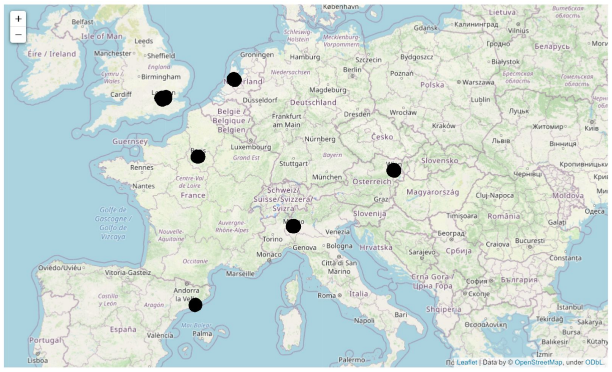


Figure 1 Distribution of Hotels in Map

From the map, we can observe that most of the hotels being reviewed are in major European cities, such as London, Paris, Milan, etc. Also, we are able to extract the countries (missing from the dataset) of the reviewed hotels: UK, Spain, Italy, France, Netherlands, and Austria. Although the map information doesn't offer much information later on for our predictive task; however, it is crucial in offering the general information. For example, if the hotels of the dataset are located in less populated areas, it would not be as representative or insightful for general travelers.

2.2.2. Feature Statistics

Before casting the numbers into labels, we must understand the general distribution of the ratings.

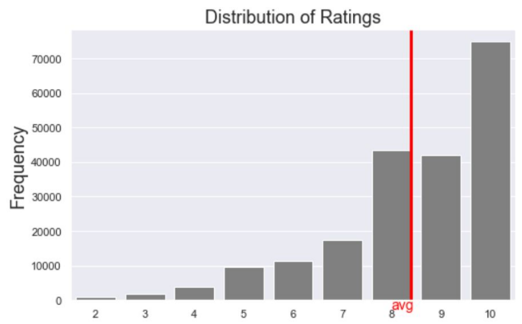


Figure 2 Distribution of Ratings in Bar Plot

Based on the graph, we can see that the distribution of hotel ratings are left skewed which means most of the people are ‘nice’ reviewers and give relatively high ratings. The average rating we calculated as the red line is about 8.39. This allows us to determine later on for our categorical labels: the average serves as a good middle line of good vs. bad user experience.

In the dataset, there is a special column called Tags, which describes the characteristics of the booking. For example, users may tag the booking as leisure or business, the amount of time they stayed at the hotel, type of the room, etc.

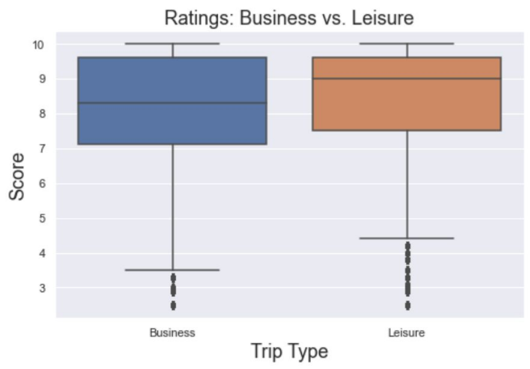
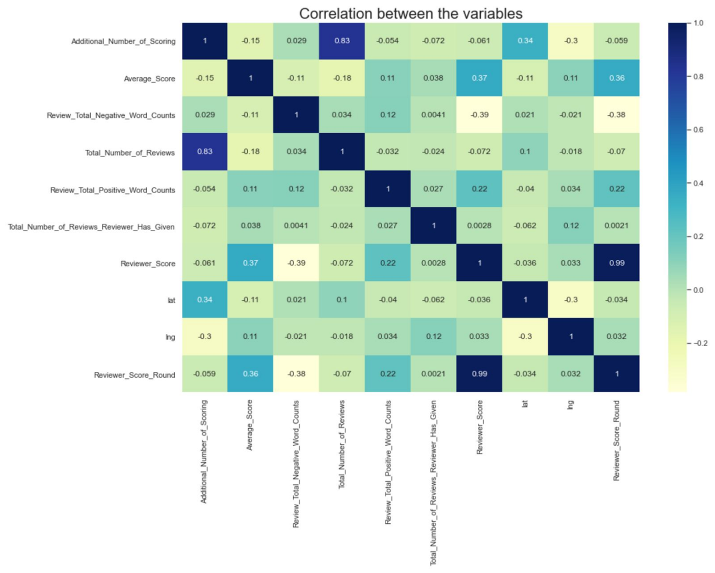


Figure 3 Distribution of Rating (Trip type) in Box Plot

From the figure above, we can see that for different types of trips, users tend to rate the hotel differently. For people booking hotels for Leisure, they tend to rate the hotel with higher scores; for people booking for business, they tend to give lower ratings. This finding could aid us to find features that are indicative of our predictive task.

After explorating the obvious features from the data, we can apply the correlation functions to all the variables with respect to the user review scores.



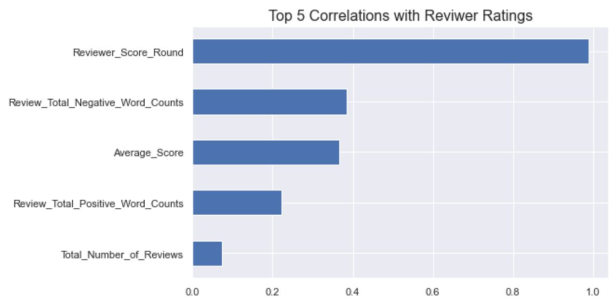


Figure 4 Feature Correlations

As seen from the heat map and bar plot above, the Reviewer_Score is most correlated with Average_Score of the hotel, Review Total Positive Word Counts, and Review Total Negative Word Counts. This makes sense as we can imagine that the more positive words a user writes tend to make him more likely to give a more satisfied rating. The same logic applies to the negative word counts.

Since our prediction features will be heavily dependent on the positive and negative reviews that a user gives to the hotel, so it would be nice to observe the words coming from both reviews, more specifically, the wordcloud of positive and negative reviews, so we can get a better understanding of the most frequently used words in each review section.

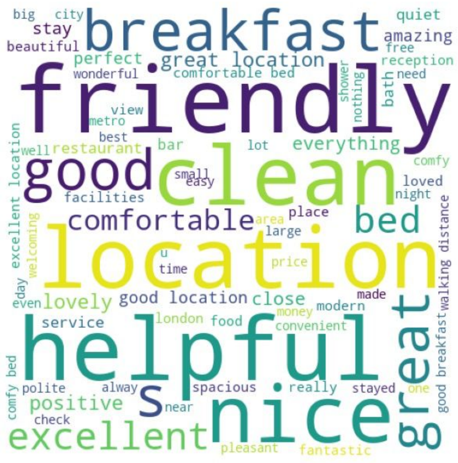


Figure 5 WordCloud of Positive_Review

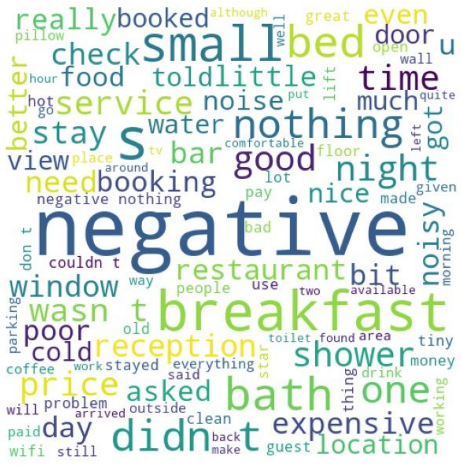


Figure 6 WordCloud of Negative_Review

From the two wordcloud images, we can note that words in the positive reviews consist of many positive adjectives describing the hotel. For example, we can relate “friendly” to the staff in the hotel, “clean” to the rooms in the hotel, “location” to the geography of the hotel, and etc. On the other hand, we can see that many of the words in the negative review wordcloud are nouns describing a particular place/element of the hotel. For example, we can relate “breakfast” to the poor breakfast that the hotel serves, “expensive” to the price of the hotel, “bath” to the bathroom conditioning of the hotel, and etc.

3. PREDICTIVE TASK

3.1 Description & Motivation of the Predictive Task

Our predictive task is to determine a user’s satisfaction level based on their review texts.

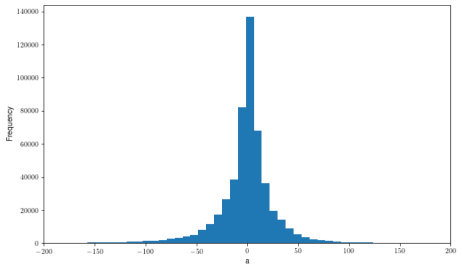
Since from the fields of the dataset, we don’t have an exact field for whether the user is feeling satisfied, neutral, or dissatisfied for a particular hotel, so we decided to use the field Review_Score to extract the three labels. We first observe the distribution of the ratings and find out the average rating is (8.4). Since the data is highly left skewed, which means the majority is satisfied with their hotel booking, we decided not to use average as the threshold, but instead we used 50 percentile and 25 percentile scores for the neutral and dissatisfied threshold. Therefore, we set the satisfaction levels to be dissatisfied (give a score of equal or lower than 7.5), neutral (give a score of equal or lower than 8.8), or satisfied (give a score of equal or lower than 10).

We chose to do such a classification task, because we think it will facilitate people with processing the text reviews and help them make better decisions. For example, some websites may provide fraud ratings in favor of benefits from the hotel company. (Many sites like ‘best … in 2020’ are all paid ads displayed from search engines.) To avoid fraudulent recommendation, we decided to implement a model that helps predict user satisfaction levels based on text reviews. To do this, we will divide the dataset into training and validation sets and evaluate the models by their accuracy and compare accuracy on the validation set and test set between different models to obtain the best model.

3.2 Baseline Model

Our **baseline model** will be a predictor that will predict based on a calculated value α:

$$\alpha = \#\, words\ in\ positive\ reviews\ -\ \#words\ in\ negative\ reviews$$



The figure above shows the distribution of the value α. The middle 25 percentile range is calculated to be [-3, 5]. From this, we will set the threshold for our baseline predictor. (We picked a 25 percentile range to correspond with our initial label mapping which was also 25 percentile range assigned to ‘neutral’.)

The model will predict satisfied if $\alpha > 5$; dissatisfied if $\alpha < -3$; neutral if $-3 \leq \alpha \geq 5$. We suppose this should have an accuracy over 50% since users tend to write longer reviews in that category when feeling satisfied/dissatisfied.

3.3 Model Selection and Comparisons

We decide to use Logistic Regression because we would like to predict an outcome variable (satisfaction level) that is categorical from predictor variables that are continuous and/or categorical. Also, since the data we are handling mostly text, tree classification methods would have a larger risk of overfitting than Logistic Regression.

For models and model comparison, we will focus most of our efforts on NLP, with some add-on features from the dataset in the final model. We will be using the baseline model for the reasoning mentioned above, a general Bag of Words Count Model, a fine-tuned Bag of Words with TFIDF Model which uses unigram and bigram respectively, a more complex model using Doc2Vec and NLTK, and finally a combined model that takes in the consideration of the weaknesses from the previous ones. We picked these models because they are popular and efficient for text classifications. We should expect models to improve following the order as above. The specifications and reasonings in details are described in the Model Section (4. Model).

3.4 Feature Extraction

Besides the use of text features, we will be using high correlated features found from EDA (figure 4): counts of positive/negative reviews, average of the hotel ratings. Since they are highly correlated with the reviewer scores, we should expect these features to be indicative in our model.

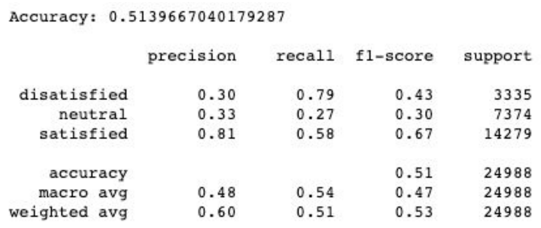
In terms of feature engineering, we will use Tag that separates reviewers into business purposes and non-business purposes. (Leisure types tend to rate higher than business purposes.) We will also be using Nltk and Word2Vec modules to generate sentimental scores and Doc2Vec vectors to represent the direct and indirect relationship among the texts and documents.

4. MODEL

4.1 Baseline Model

The baseline we will use for comparison is a well-constructed naive model doing a "Difference in Word Count" approach by using the difference between the number of words in a positive review and negative review to determine the user satisfaction level. The reason why this baseline model is appropriate and meaningful because intuitively, users would tend to write longer review (for either positive or negative) if the user feels either satisfied or dissatisfied for the hotel, therefore by using the selected threshold α from above and the difference between positive review word counts and negative review word counts, we can get a basic sense on whether a user would feel a certain way (satisfaction level) toward the hotel.

Below is the result on accuracy on the validation set.



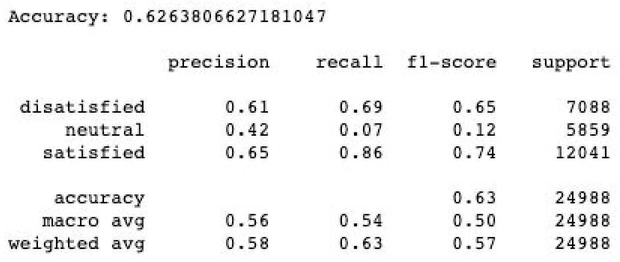
```
Accuracy: 0.5139667040179287

              precision    recall  f1-score   support

 disatisfied       0.30      0.79      0.43      3335
     neutral       0.33      0.27      0.30      7374
   satisfied       0.81      0.58      0.67     14279

    accuracy                           0.51     24988
   macro avg       0.48      0.54      0.47     24988
weighted avg       0.60      0.51      0.53     24988
```

4.2 Bag of Words Model

We first try the bag of words approach by k number of most frequent words from the Positive_Review section and Negative_Review section in the train dataset. Then, we combine those $2k$ words into a matrix and do a one-hot encoding on counting how many frequent words appeared in the positive and negative reviews. After that, we train our logistic regression model using the one-hot encoding feature as input and labels as output and we do prediction on the validation set.

The reason why we choose to use the positive and negative reviews is that from the wordcloud above, we notice that words in the positive and negative review are quite different and can potentially help predict the user experience solely based on the most frequent words. Also, the reason why we start off with the Bag of Words model is because the approach is very simple to implement and is very commonly used in methods of text and document classification.

To optimize the Bag of Words model, we essentially used grid search to find the model that gives the best accuracy by iterating over the regularizer and the number of words used for each review, and finally we obtain 1000 most frequent words from positive and negative reviews, respectively, and a logistic regression model with a regularizer of C=10.0. The accuracy we achieved is roughly 0.6264.



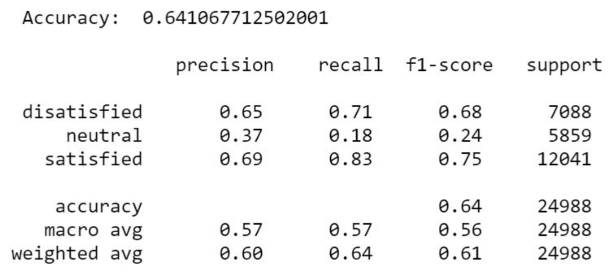
```
Accuracy: 0.6263806627181047

              precision    recall  f1-score   support

 disatisfied       0.61      0.69      0.65      7088
     neutral       0.42      0.07      0.12      5859
   satisfied       0.65      0.86      0.74     12041

    accuracy                           0.63     24988
   macro avg       0.56      0.54      0.50     24988
weighted avg       0.58      0.63      0.57     24988
```

4.3 BoW with TF-IDF Model

4.3.1 Unigram

To improve the baseline model, we first used the bag-of-words method to represent dataset as a matrix of frequency of most common words, then we resorted to TF-IDF to evaluate the relevance of keywords to specific documents and use these term frequency - inverse document frequency as numeric statistics to represent keywords in every document. TF-IDF increases proportionally to the number of times a word appears in the document, offset by the number of documents in the corpus that contain the word. Since The TF-IDF Model chooses the more informative words and those keywords could provide context for specific documents, rather than most common words across the entire documents, we expect the performance of the tf-idf model would beat at least the baseline model.

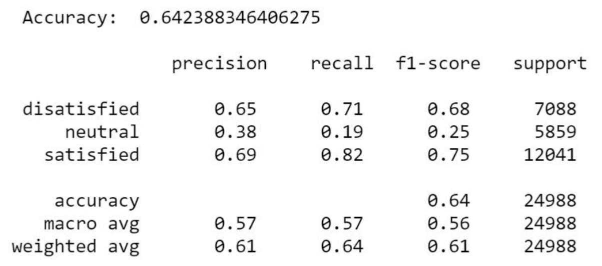
We incorporated TF-IDF transformed documents matrix along with the average of the given hotel as features into the logistic regression model, and we conducted a Gridsearch for the best performing regularization parameter and the number of keywords to run the tf-idf method on. We find out that with the unigram tf-idf, the best performing model is to exclude stopwords in English and remove all the punctuations and use 5000 keywords for tf-idf. The regularization parameter performs the best is C=1.0. We achieved 0.641 overall accuracy for this model.



```
Accuracy:  0.641067712502001

              precision    recall  f1-score   support

disatisfied        0.65      0.71      0.68      7088
    neutral        0.37      0.18      0.24      5859
  satisfied        0.69      0.83      0.75     12041

   accuracy                            0.64     24988
  macro avg        0.57      0.57      0.56     24988
weighted avg       0.60      0.64      0.61     24988
```

4.3.2 Bigram

Unigram method may result in not differentiating between words in negative context and words in positive context, such as “recommend” versus “not recommend”. To better understand the context of keywords for documents, we implemented bigram as alternative features for our logistic regression model. We use gridsearch to find parameters for this model. And the best performing parameter found by gridsearch is C=10 and 5000 uni&bigrams. The accuracy we achieved is 0.642.



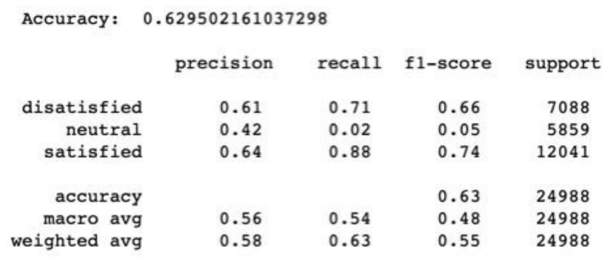
```
Accuracy:  0.642388346406275

              precision    recall  f1-score   support

disatisfied        0.65      0.71      0.68      7088
    neutral        0.38      0.19      0.25      5859
  satisfied        0.69      0.82      0.75     12041

   accuracy                            0.64     24988
  macro avg        0.57      0.57      0.56     24988
weighted avg       0.61      0.64      0.61     24988
```

4.4 Gensim(Doc2Vec) with NLTK(SIA)

The last model we worked on used gensim Word2Vec modules. More specifically, we used the extended version of Word2Vec: Doc2Vec. While Word2Vec captures the relationship among words, Doc2Vec adds on more information by also capturing the relationships among documents. Application wise, Word2Vec maps semantic relations to vectors that capture word similarities. However, words relationships are limited in our case when we try to find relationships/similarities between satisfied/neutral/negative reviews. That is, we need relationships between reviews rather than words. Therefore, we applied Doc2Vec rather than Word2Vec here.

Besides using Doc2Vec, we would like to capture the sentiments directly rather than using BoW and ask the Classifier to figure itself out. So, this is when we applied SentimentIntensifierAnalyzer. This package from NLTK uses a rich dictionary to map the positive/neutral/negative words into numerical values. Afterall, it also produces a compound score that captures the overall sentiment of the review.

Again, we used Logistic Regression here and applied Grid Search get the optimized result on the validation set:



```
Accuracy:  0.629502161037298

              precision    recall  f1-score   support

disatisfied        0.61      0.71      0.66      7088
    neutral        0.42      0.02      0.05      5859
  satisfied        0.64      0.88      0.74     12041

   accuracy                            0.63     24988
  macro avg        0.56      0.54      0.48     24988
weighted avg       0.58      0.63      0.55     24988
```

4.5 Limitations and Problems

For **scalability**, we do face a potential threat that if our data gets much larger, the BoW model will be very computationally expensive. To possibly resolve this problem, we may have to set a threshold for the max dictionary size relative to the amount of data.

For **overfitting**, we will tune our parameters C according to the validation set, so that the model will try to avoid the issue of overfitting by applying an appropriate regularizer.

For **models with BoW**, it’s very time consuming to train the model, especially with a large dictionary size. We were able to solve this problem by converting the matrix into a sparse matrix so that it speeds up the process of fitting the model.

One **failure attempt** we had was trying to fit the BoW (with TF IDF) data with the Support Vector Machine Learning model (sklearn.svm.SVC). We ended up getting a low accuracy score. We suppose this happened because our dataset does not have a clear margin that separates the classes so that SVC may not be the appropriate model here.

For our **model using Doc2Vec and Nltk**, we were expecting a high accuracy because it accounts for the sentiment in the text directly instead of making the model learn itself. However, the accuracy of the model was close to our original BoW model. However, we do notice that the recall for this model is really high for dissatisfied and satisfied classifications. But the recall was nearly zero for the neutral classification. We suppose that this model can distinguish between the two extremes of the sentiments well but less sensitive to the neutral sentiments. This makes sense as we can refer back to the word cloud that words like ‘excellent, clean, helpful, comfortable…’ or ‘didn’t, nothing, expensive, small’ are strong sentimental words that appear often in the review texts.

4.6 Final Model

Afterall, we decided to combine our last model with the TF IDF model in the expectation that the TF IDF model could compensate for the limitation in classifying the 'neutral' categories. In our final model, we also utilized the features which had high correlation with the review scores from EDA: Average_Score of the hotel, Review Total Positive Word Counts, and Review Total Negative Word Counts. We also added in the engineered feature: 'isBusiness' (whether the trip was booked for business or leisure).

Here is the final result reported on the validation set from our model tuning:

```
Accuracy:  0.6649591804065952

              precision    recall  f1-score   support

 disatisfied       0.67      0.76      0.71      7088
     neutral       0.42      0.14      0.21      5859
   satisfied       0.70      0.87      0.77     12041

    accuracy                           0.66     24988
   macro avg       0.59      0.59      0.56     24988
weighted avg       0.62      0.66      0.62     24988
```

5. LITERATURE

The dataset we used for hotel reviews is from Kaggle (https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe) and is scraped from Booking.com. The dataset contains 515,000 customer reviews and scoring of 1493 luxury hotels across Europe, the geographical location of hotels are also provided for further analysis. From the Notebooks section, we notice that the dataset is mostly used for sentiment analysis and a few visualization analysis.

Other similar datasets on predicting from the hotel reviews include datasets scraped from hotel booking websites such as http://www.tripavdisor.com, http://www.agoda.com. Wei-Ta Chu and Wei-Han Huang analyzed visual features such as hotel's cover photo, and investigated the relationship between hotel ratings and visual information and how incorporation of cultural difference and visual information could improve the prediction model[1]. Yu-ning Xiong and Li-xiao Geng also used datasets scraped from online hotel reservation websites to construct a personalized hotel recommendation system based on consumer purchasing history and hotel information[2].

In the field of language model, much work has been done to better classify sentiments of words. Jenq-Haur Wang, Ting-Wei Liu, and Xiong Luo Long Wang adopted LSTM along with word2Vec to classify word sentiments. LSTM is a unique form of RNN and it could store context of the words in features. LSTM continuously updates the model weights through each epoch until the loss is determined to be minimized[3].

In comparison with the word2vec and LSTM method[3], our final model also incorporates word2Vec. Instead of incorporating word2vec into LSTM as [3] did, we directly put the word2Vec values along with sentimental insentifier (Nltk) into our logistic regression model. We used gridsearch for the best performing parameters for logistic regression model, rather than continuously updating the weights vectors assigned to features during each epoch as in LSTM. For the conclusion part, we agreed/verified in our work that word2Vec as a word-embedding model could feasibly train the contextual semantics of words in short context [3]. In addition, [3] also utilized the state-of-the-art method LSTM to reduce loss during training, which we admit that LSTM as a deep learning method could outperform our Logistic model when the training set is large.

6. RESULTS AND CONCLUSIONS

Below is a table that combines and contrasts the performance of all our models using accuracy as metric and run on validation set and test set:

Model	Validation Accuracy	Test Accuracy
Baseline model	51.39%	50.46%
Bag of words	62.63%	61.17%
Tf-idf unigram	64.10%	62.21%
Tf-idf bigram	64.23%	63.49%
Gensim(Doc2Vec) with NLTK(SIA)	62.95%	62.03%
Final model	66.50%	66.96%

As shown from the above figure, our final model outperforms all alternative models. This is an expected result. The baseline model would classify most reviews as satisfied reviews and ignore the situation where consumers write equal length of positive and negative reviews. Bag of words did not consider words that appeared less frequent but could be strongly indicative of sentiment. Unigram tf-idf ignores that word context could inverse the sentiment of reviews. Bigram tf-idf treats words with negative and positive connotations indifferently, so its accuracy could be further improved. The Gensim and NLTK model performs well on words with strong sentiments, but could be improved in classifying neutral review, and from our trial we get that if the classification is binary, Gensim could have performed significantly better than previous models.

Our final model that incorporates tf-idf, word2Vec and features in datasets such as Average_Score of the hotel, Review Total Positive Word Counts, and Review Total Negative Word Counts combines word2Vec features that best summarized words with strong negative or positive sentiments with tf-idf features that could better represent neutral words and keywords to specific documents, and they along with features extracted directly from the dataset could provide a more representative description of both the negative and positive review words. Below is a detailed accuracy report of test accuracy of the final model:

```
Accuracy:  0.6695936387140836

              precision    recall  f1-score   support

 disatisfied       0.67      0.74      0.71      5876
     neutral       0.45      0.16      0.24      4786
   satisfied       0.70      0.87      0.78      9837

    accuracy                           0.67     20499
   macro avg       0.61      0.59      0.57     20499
weighted avg       0.63      0.67      0.63     20499
```

Below, we reported the **ROC curves** for all three classes. Like what we expected from the validation result, our model is very sensitive in classifying satisfied and dissatisfied sentiments within our text, the area under the curve achieved about 0.8 for both classes (figure 6, 8), which by convention, is an excellent distrimination measure. However, we do notice our shortcomings with our model: We are not good at classifying neutral satisfaction (figure 7).

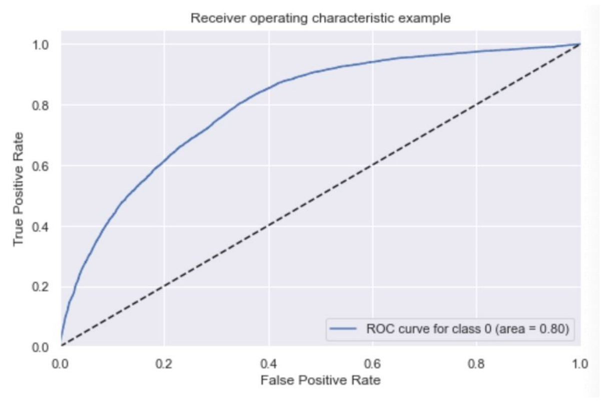


Figure 6 ROC for classifying 'satisfied'

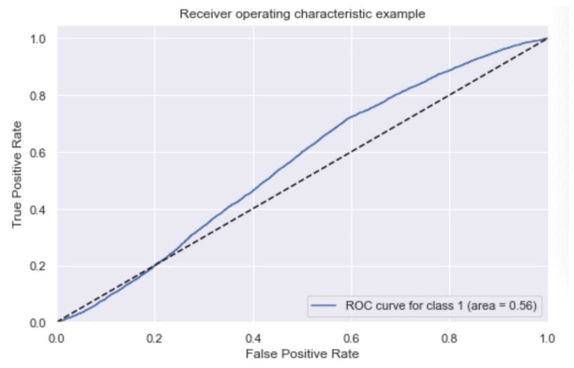


Figure 7 ROC for classifying 'neutral'

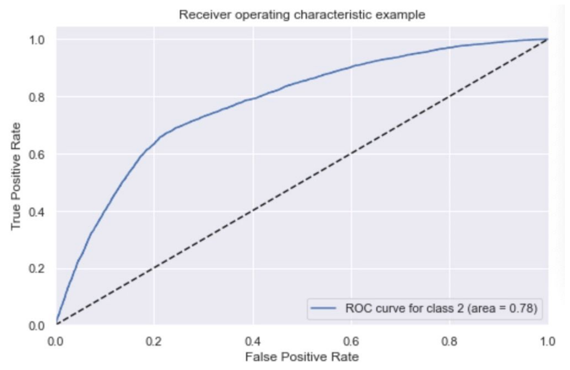


Figure 8 ROC for classifying 'dissatisfied'

In conclusion, six models were proposed and discussed, with our final model that incorporates features from previous models and features in the original dataset performs the best. We were successful in predicting the satisfied and dissatisfied satisfaction levels but weak in predicting neutral sentiments. With a better deep learning knowledge in the future, rather than using direct sentiment intensifier, our classification model could possibly be further improved by implementing state-of-the-art method BERT, as BERT reads the entire sequence of words at once, considers the context of words with previous words and context of next words, and is shown to beat most other NLP models in sentiment analysis[4].

7. REFERENCES

[1] Chu, WT., Huang, WH. Cultural difference and visual information on hotel rating prediction. World Wide Web 20, 595–619 (2017).

https://doi.org/10.1007/s11280-016-0404-2

[2] Y. Xiong and L. Geng, "Personalized Intelligent Hotel Recommendation System for Online Reservation--A Perspective of Product and User Characteristics," 2010 International Conference on Management and Service Science, Wuhan, 2010, pp. 1-5, doi: 10.1109/ICMSS.2010.5576790.

[3] Wang, JH., Liu, TW., Wang, X Luo. An LSTM Approach to Short Text Sentiment Classification with Word Embeddings. The 2018 Conference on Computational Linguistics and Speech Processing ROCLING 2018, pp. 214-223

[4] Z. Gao, A. Feng, X. Song and X. Wu, "Target-Dependent Sentiment Classification With BERT," in IEEE Access, vol. 7, pp. 154290-154299, 2019, doi: 10.1109/ACCESS.2019.2946594.