



BT4222: Mining Web Data for Business Insights

Final Project Report: Sentiment Analysis and Topic Modelling of Hotel Reviews

Members:

Liew Mei Xin (A0171703M)

Yue Jing Ying (A0171154M)

1. Motivation	4
2. Dataset	4
3. Libraries and Packages	6
4. Data Preprocessing	7
4.1 Dataframe	7
4.2 Reviews	7
5. Exploratory Data Analysis	9
5.1 Location of Hotels	9
5.2 Distribution of Hotel Average Scores	10
5.3 Top 10 Famous Hotels	11
5.4 Top 10 Positively-Rated Hotels	12
5.5 Top 10 Negatively-Rated Hotels	13
5.6 Word Cloud for Positive Reviews	14
5.7 Word Cloud for Negative Reviews	14
6. Sentiment Analysis	15
6.1 Word Embedding	15
6.2 Choice of Models	17
6.2.1 Logistic Regression	18
6.2.2 Multinomial Naive Bayes	19
6.2.3 Random Forest	20
6.2.4 Comparison of Results	21
6.3 Further Extension	22
6.3.1 Lemmatization vs Stemming	22
6.3.2 Convolutional Neural Network (CNN)	23
7. Topic Modelling	26
7.1 Define Model Parameters	26
7.2 Find Optimum Number of Topics	26
7.3 Visualise Topics and Related Keywords	27
8. Future Improvements	29
8.1 Further Sentiment Analysis Model Exploration	29
8.2 Grouped Topic Modelling	29
8.3 Single Sentiment Analysis and Topic Modelling Pipeline	30
9. Possible Extensions	31
9.1 Standardised Rating Generator	31

9.2 Hotel Recommender	31
10. Conclusion	32
11. References	33

1. Motivation

In this project, we seek to uncover how hotel reviews can be better utilised by both companies and customers to make more informed decisions. On the company's side, hotels usually collect feedback from their customers through reviews, from which they can gather areas they are doing well or poorly in. Hotels can then maintain efforts in areas customers are satisfied with while working harder to fix areas customers feel they are lacking in. On the customer's side, many tend to read hotel reviews before booking rooms for leisure or business trips to gauge which hotels offer better services than others. This can be based on ratings or reviews left by previous customers.

However, it is often difficult to make good inferences from raw hotel reviews. Ratings are subjective and may be biased. It is difficult to determine if a review is considered positive or negative. Areas of satisfaction or dissatisfaction also differ from customer to customer. As such, we wish to create a standardised positive or negative review classifier through sentiment analysis and machine learning classification. This provides customers with a more precise representation of hotel performance, making it easier for them to find out which is the best hotel to stay at. We also hope to discover the main areas majority of customers are satisfied and dissatisfied with through topic modelling. This will give hotels a clearer picture of possible course of actions to take next to improve services.

2. Dataset

The dataset we used is '*515K Hotel Reviews Data in Europe*', available on Kaggle. A link to the dataset has been provided under the References section of this report.

The dataset contains 515,738 customer reviews of 1,493 luxury hotels across Europe, scraped from Booking.com. Saved as a csv file, it contains 17 fields, namely:

- 1) *Hotel_Address*: Address of the hotel
- 2) *Additional_Number_of_Scoring*: Number of scores without reviews for the hotel
- 3) *Review_Date*: Date the review was written and posted
- 4) *Average_Score*: Average score of the hotel; considers up to the last review in the previous year
- 5) *Hotel_Name*: Name of the hotel
- 6) *Reviewer_Nationality*: Nationality of the reviewer
- 7) *Negative_Review*: Negative review given by the reviewer to the hotel. If no negative review was given, "No Negative" is assigned
- 8) *Review_Total_Negative_Word_Counts*: Total number of words in the negative review
- 9) *Total_Number_of_Reviews*: Total number of reviews the hotel has

- 10) *Positive_Review*: Positive review given by the reviewer to the hotel. If no positive review was given, "No Positive" is assigned
- 11) *Review_Total_Positive_Word_Counts*: Total number of words in the positive review
- 12) *Total_Number_of_Reviews_Reviewer_Has_Given*: Total number of reviews the reviewer has given before
- 13) *Reviewer_Score*: Score the reviewer gave to the hotel, based on his/her experience
- 14) *Tags*: Tags the reviewer gave for the hotel
- 15) *days_since_review*: Number of days from the day the review was posted to the day the review was scraped
- 16) *lat*: Latitude of the hotel's location
- 17) *lng*: Longitude of the hotel's location

Figures 1-3 below show screenshots of the dataset:

	Hotel_Address	Additional_Number_of_Scoring	Review_Date	Average_Score	Hotel_Name	Reviewer_Nationality	Negative_Review
0	Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	8/3/2017	7.7	Hotel Arena	Russia	I am so angry that i made this post available...
1	Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	8/3/2017	7.7	Hotel Arena	Ireland	No Negative
2	Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	7/31/2017	7.7	Hotel Arena	Australia	Rooms are nice but for elderly a bit difficul...

Review_Total_Negative_Word_Counts	Total_Number_of_Reviews	Positive_Review	Review_Total_Positive_Word_Counts	Total_Number_of_Reviews_Reviewer_Has_Given
397	1403	Only the park outside of the hotel was beauti...	11	7
0	1403	No real complaints the hotel was great ...	105	7
42	1403	Location was good and staff were ok It is cut...	21	9

Reviewer_Score	Tags	days_since_review	lat	lng
2.9	['Leisure trip ', 'Couple ', 'Duplex Double...']	0 days	52.360576	4.915968
7.5	['Leisure trip ', 'Couple ', 'Duplex Double...']	0 days	52.360576	4.915968
7.1	['Leisure trip ', 'Family with young childre...']	3 days	52.360576	4.915968

Figures 1-3: Screenshots of Dataset

The creator of the dataset has already removed unicode and punctuations from the text data in *Negative_Review* and *Positive_Review*.

3. Libraries and Packages

The libraries and packages we deployed include:

- 1) pandas: For data structuring
- 2) NumPy: For scientific computing and data manipulation
- 3) spaCy: For lemmatization
- 4) Gensim: For topic modelling
- 5) NLTK: For natural language processing
- 6) scikit-learn: For classification, regression and model selection
- 7) Matplotlib: For data visualisation
- 8) Seaborn: For data visualisation
- 9) Folium: For map visualisation
- 10) Wordcloud: For word cloud generation

4. Data Preprocessing

4.1 Dataframe

After importing the csv file as a pandas dataframe, we checked and found 526 reviews that are duplicates, so we removed them. Next, we checked and found 3,268 reviews with missing *lat* and *lng* values. Rather than simply removing these reviews, we chose to use hotel addresses to retrieve the missing latitude and longitude values from <https://www.latlong.net/>, minimising loss of information. Lastly, we removed columns that are unnecessary in our analysis, leaving only 9 columns: *Average_Score*, *Hotel_Name*, *Reviewer_Nationality*, *Negative_Review*, *Positive_Review*, *Reviewer_Score*, *Tags*, *lat* and *lng*. A screenshot of the dataset after cleaning is shown in Figure 4 below:

	Average_Score	Hotel_Name	Reviewer_Nationality	Negative_Review	Positive_Review	Reviewer_Score	Tags	lat	lng
0	7.7	Hotel Arena	Russia	I am so angry that i made this post available...	Only the park outside of the hotel was beauti...	2.9	['Leisure trip', 'Couple', 'Duplex Double...]	52.360576	4.915968
1	7.7	Hotel Arena	Ireland	No Negative	No real complaints the hotel was great great ...	7.5	['Leisure trip', 'Couple', 'Duplex Double...]	52.360576	4.915968
2	7.7	Hotel Arena	Australia	Rooms are nice but for elderly a bit difficul...	Location was good and staff were ok It is cut...	7.1	['Leisure trip', 'Family with young childre...]	52.360576	4.915968
3	7.7	Hotel Arena	United Kingdom	My room was dirty and I was afraid to walk ba...	Great location in nice surroundings the bar a...	3.8	['Leisure trip', 'Solo traveler', 'Duplex...]	52.360576	4.915968
4	7.7	Hotel Arena	New Zealand	You When I booked with your company on line y...	Amazing location and building Romantic setting	6.7	['Leisure trip', 'Couple', 'Suite', 'St...]	52.360576	4.915968

Figure 4: Screenshot of Dataset after Cleaning

4.2 Reviews

To prepare the reviews for sentiment analysis and topic modelling, we extracted the text data in *Positive_Review* and *Negative_Review* and saved them as *pos_reviews* and *neg_reviews* lists respectively. We also created an *all_reviews* list - a combination of *pos_reviews* and *neg_reviews* - which contains all reviews, regardless of whether they are positive or negative. We then performed text cleaning, which is an important process for improving performance of text classification and topic modelling models:

Lowercasing

Convert all uppercase letters to lowercase for consistency and ease of processing using Gensim's `simple_preprocess` function. For example, the sentence "Very Comfortable Bed and Cool Hotel" will become "very comfortable bed and cool hotel".

Tokenization

Break sentences up into its individual words using Gensim's `simple_preprocess` function. For example, the sentence "very comfortable bed and cool hotel" will be tokenized to ['very', 'comfortable', 'bed', 'and', 'cool', 'hotel']. This is so that the importance of each word can be determined based on its frequency.

Stop Words Removal

Stop words are commonly occurring words such as "I", "a", "is" and "the", which are meaningless and should be filtered out. This is done by comparing tokens to a list of stop words from the NLTK library. If a token is not found in the list of stop words, it is not a stop word and thus, should be taken. Otherwise, it is a stop word and should be left out. For example, after removal of stop words, ['very', 'comfortable', 'bed', 'and', 'cool', 'hotel'] will become ['comfortable', 'bed', 'cool', 'hotel']. By removing low information words, we can focus on more important words instead.

Stemming and Lemmatization

Since the dataset is very large (>1 million rows), stemming is done to remove prefixes and suffixes from words in the reviews. Even though some reduced words may not be actual words in the dictionary (e.g. "troubling" to "troubl"), stemming is faster than lemmatization. However, as we explored how accuracy of classification models differs between different preprocessing techniques as well as usage of topic modelling, lemmatization is also used on original reviews to reduce inflected words to their root forms. This will be further elaborated in the later parts of this report. For example, 'helps', 'helping', 'helped' will all be lemmatized to 'help'. This is necessary to ensure that words with the same root form are not interpreted as different words. We wrote a function to handle lemmatization with the use of spaCy's 'en' model, where we only kept tokens with 'NOUN', 'ADJ', 'VERB' and 'ADV' part-of-speech tags. This is because such words give better information and thus, are more meaningful.

Short Words Removal

Remove words that are less than 3 characters in length as it is highly unlikely that such short words are significant.

5. Exploratory Data Analysis

5.1 Location of Hotels



Figure 5: Map of Hotel Locations

From Figure 5 above, hotels in the dataset are located in the European region: United Kingdom, Netherlands, France, Italy, Spain and Austria.

5.2 Distribution of Hotel Average Scores

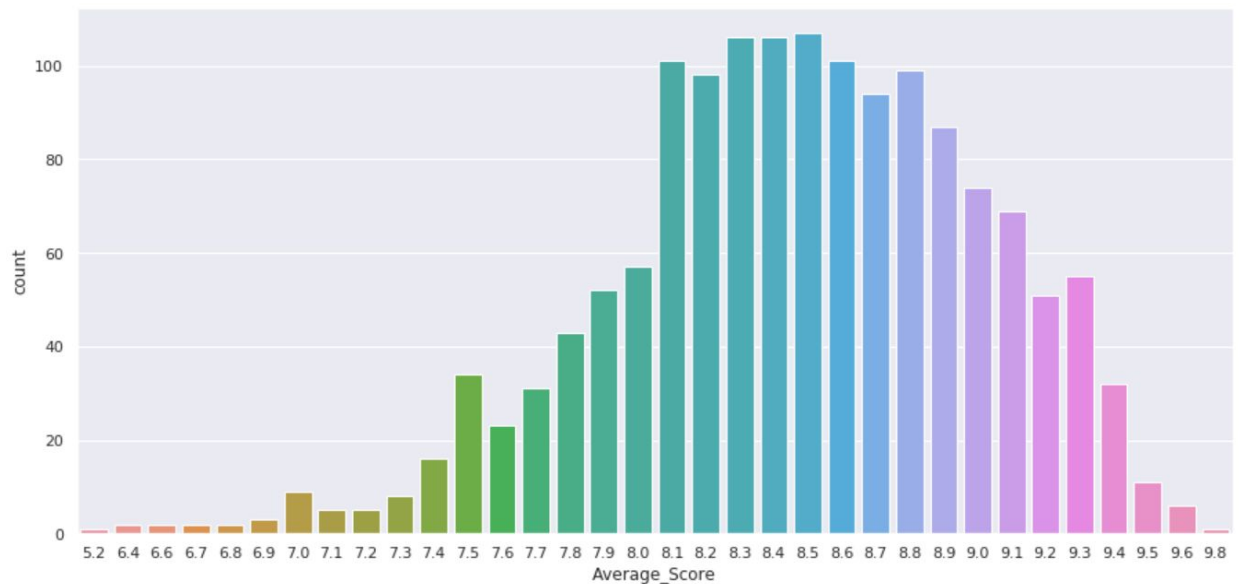


Figure 6: Graph of Hotel Average Scores

To get a sense of how the hotels are generally performing, we plotted a graph to visualise the distribution of *Average_Score* values across all hotels. From Figure 6 above, we can observe that most *Average_Score* values lie in the range of 8.1 to 8.8. We also noticed that the scores are at least 5.2 and at most 9.8. Since the scores are not absolute, it could mean that most of the reviews given by users are not absolutely at positive or negative extreme ends, but could be a mix of both positive and negative aspects. Therefore, the dataset given has filtered sections of reviews belonging to positive or negative classes.

5.3 Top 10 Famous Hotels

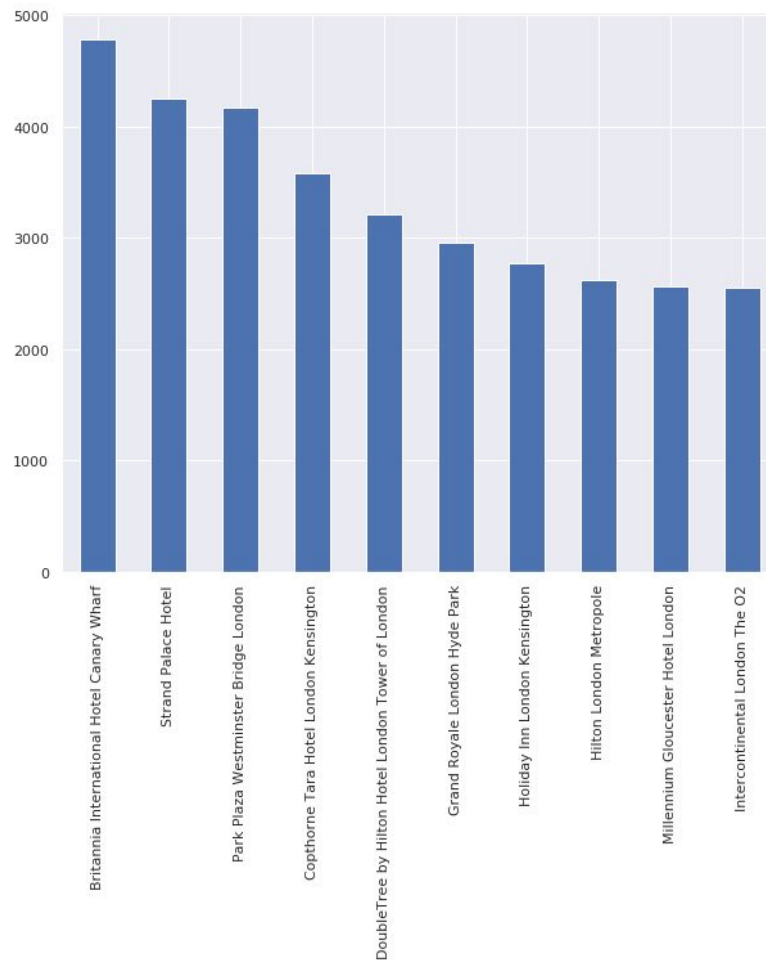


Figure 7: Graph of Top 10 Famous Hotels

We determined the fame of a hotel based on the total number of reviews it has. From Figure 7 above, we can infer that Britannia International Hotel Canary Wharf is the most famous as it received the greatest number of reviews from past customers. This information can make deciding on a place to stay at more convenient for potential customers. Customers tend to choose from hotels with many reviews as they feel more safe as compared to choosing hotels with minimal information available.

5.4 Top 10 Positively-Rated Hotels

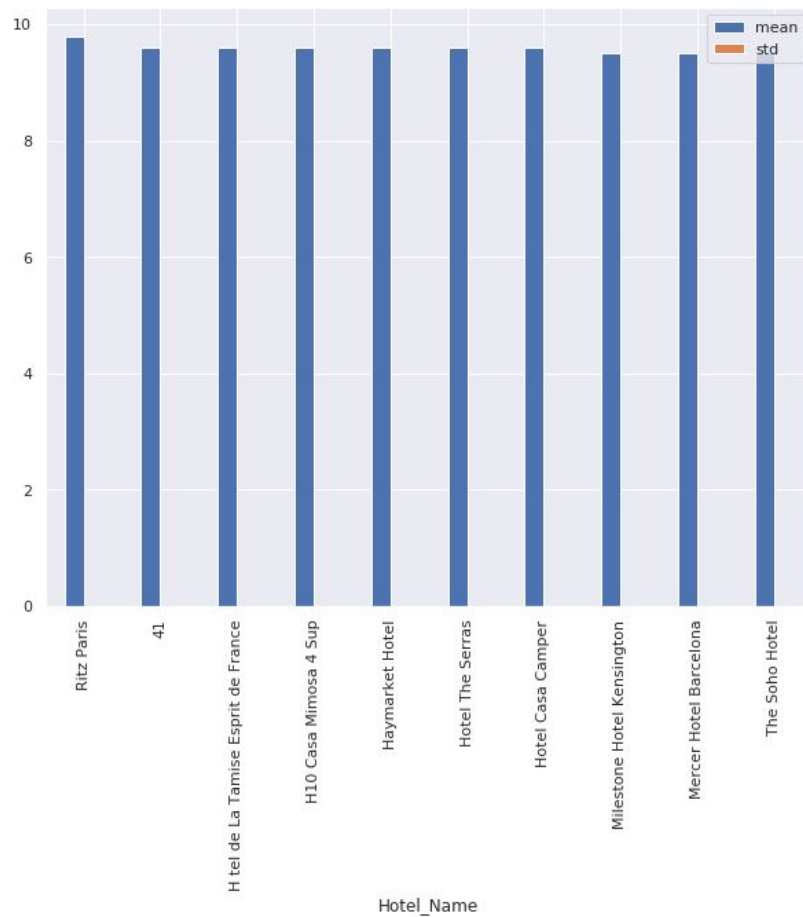


Figure 8: Graph of Top 10 Positively-Rated Hotels

We counted the total number of positive reviews each hotel received and sieved out the top 10 with the most of such reviews. From Figure 8 above, we can tell that Ritz Paris received the greatest number of positive reviews. However, we also noted that the difference in the numbers between the top 10 hotels are very similar since the heights of the bars do not differ by much. This information will come in handy for potential customers as they are keen on staying at hotels that leave good impressions rather than ones where past customers deemed as subpar.

5.5 Top 10 Negatively-Rated Hotels

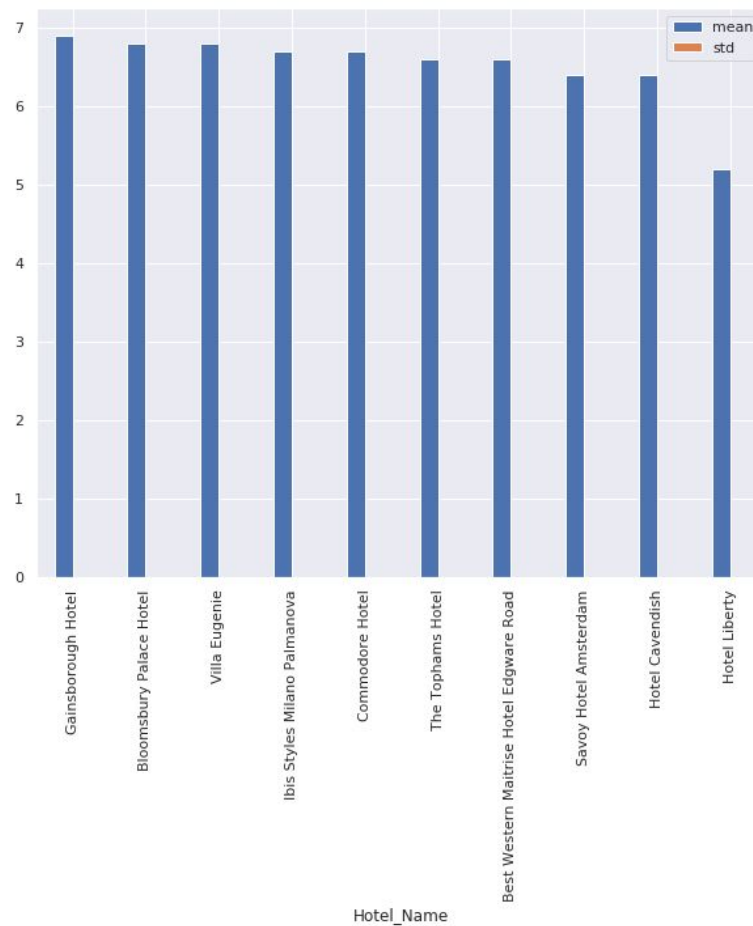


Figure 9: Graph of Top 10 Negatively-Rated Hotels

We also counted the total number of negative reviews each hotel received and filtered out the top 10 with the most of such reviews. From Figure 9 above, we find that Gainsborough Hotel received the greatest number of negative reviews. The numbers are still quite close this time but with more variance than that of positive reviews. This information is highly relevant to potential customers as they definitely will not want to stay at hotels that are lacklustre. As such, they can consider disregarding these hotels when deciding where to stay at.

5.6 Word Cloud for Positive Reviews



Figure 10: Word Cloud for Positive Reviews

We generated a word cloud, as seen in Figure 10 above, for words that appeared in all the positive reviews. While this allows us to pick out what customers were happy with, such as “location”, “staff” or “breakfast”, these words all belong to different topics. It is unclear which words relate to which topic.

5.7 Word Cloud for Negative Reviews



Figure 11: Word Cloud for Negative Reviews

We generated another word cloud, as seen in Figure 11 above, for words that appeared in all the negative reviews. Again, while this allows us to pick out what customers were unhappy with, such as “price”, “room” or “noise”, these words also belong to different topics. Just like the case for positive reviews, it is unclear which words relate to which topic, making it difficult for hotels to pinpoint what exactly do they have to fix. Hence, we were incentivised to conduct topic modelling to divide all the words into topics and their related keywords for easier analysis.

6. Sentiment Analysis

6.1 Word Embedding

After processing the reviews, each word in the review has to be encoded as integers or floating point values to be used as an input to machine learning algorithms. This process is called Word Embedding. We used the following methods to conduct word embedding:

- Bag of Words (BOW)
- Term Frequency-Inverse Document Frequency (TF-IDF)

BOW measures the number of words in each document. To use the BOW model, there is an in-built function in scikit-learn machine learning library in Python called “CountVectorizer()”. `CountVectorizer.fit(X_train).vocabulary_()` gives the index of the word:

```
{'park': 42313, 'hotel': 28232, 'beautiful': 5744, 'real': 46896, 'complaint': 12362}
```

As for TF-IDF, it consists of two parts: TF (Term Frequency) which is similar to the output of BOW and follows the formula: $TF = \frac{\text{Frequency of a term in a document}}{\text{Total number of terms in all documents}}$. Inverse Document Frequency (IDF) measures the weight of rare words across all documents. The more times a word appears in documents, the less “significant” the word becomes. $IDF = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents with term } t}\right)$.

idf_weights	
room	2.416122
staff	2.537503
the	2.633417
no	2.690008
location	2.844148

Figure 11: Low IDF Weights

idf_weights	
made	5.162080
poor	5.181331
easy	5.192573
near	5.197403
around	5.198630

Figure 12: High IDF Weights

From Figures 11 and 12 above, we can see that the words “room”, “staff” and “the” have low IDF weights, which means they are common words in most reviews whereas in comparison, the words “poor” and “easy” have high IDF weights, which means they are less common in most reviews.

Hence, TF-IDF is the product of the TF and IDF frequencies, measuring how important a word is to a document. For a word to have a high TF-IDF, it must appear frequently in a document and be absent in other documents, so that it can become the signature word of that document.

	amsterdam	anyone	aside	back	background	bit	check	checkout	complaints
no real complaints hotel great great location surroundings rooms amenities service two recommendations however firstly staff upon check confusing regarding deposit payments staff offer upon checkout refund original payment make new one bit confusing secondly site restaurant bit lacking well thought excellent quality food anyone vegetarian vegan background even wrap toasted sandwich option would great aside minor minor things fantastic spot back return amsterdam	0.111803	0.111803	0.111803	0.111803	0.111803	0.223607	0.111803	0.111803	0.111803

Figure 13: TF-IDF Score of a Review Sample

From Figure 13 above, the word “bit” has a higher TF-IDF score than other words in the review, which means it is less common in other reviews and more unique to this particular review sample.

However, these above two methods (BOW and TF-IDF) do not take into account the sequence of words in a sentence. The order of words in a sentence does matter a lot as they determine the meaning of the sentence. Different orders may result in different meanings even though they use the same words. For example, “the hotel guests take room card from manager” versus “the manager take room card from hotel guests” have different meanings due to different word sequences. Therefore, we want to make use of n-grams in BOW and TF-IDF respectively for optimal word vectorization. N-gram represents n-consecutive words in a sentence. For example, Figure 14 below shows TF-IDF bigram on a review sample, which measures TF-IDF values of 2 consecutive words in a sentence:

	bigram	tfidf values
0	no real	0.117
1	real complaints	0.230
2	complaints hotel	0.131

Figure 14: TF-IDF Bigram on a Review Sample

There are different combinations of n-grams that can be used in both BOW and TF-IDF. Since different combinations of word embedding models can result in different accuracies, we want to measure the accuracy of each combination in the following models - Logistic Regression, Multinomial Naive Bayes and Random Forest - to determine which combination is optimal to derive the highest accuracy. The dataset is first split into training and test by ratio of 8:2. Each combination of word vectorization will be fed into the algorithm to train on training dataset, and then test its accuracy on test dataset. The table in Figure 15 below shows the most optimal word embedding combination for each classification model, with the number of features on the x-axis, which represents the top words ordered by top frequency across the corpus. If the

number of features is 100,000, then the TF-IDF will select the top 100,000 words that most frequently appear.

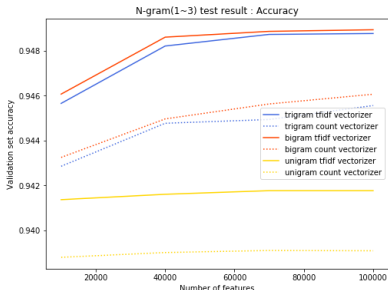
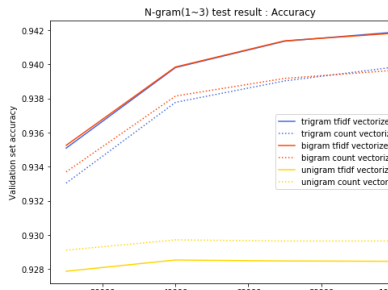
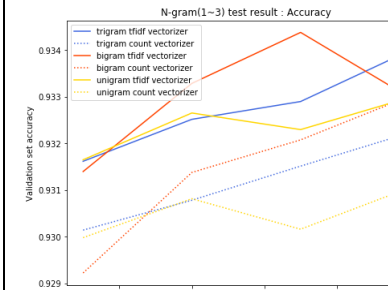
Logistic Regression	Multinomial Naive Bayes	Random Forest
		
Best model: TFIDF Bigram (100,000 word features)	Best model: TFIDF Trigram (100,000 word features)	Best model: TFIDF Bigram (70,000 word features)

Figure 15: Table of Most Optimal Word Embedding Combination

6.2 Choice of Models

After removal of null reviews, there is a total of 1,030,424 reviews and 80,112 unique words, of which 515,212 reviews belong to positive and negative labels respectively. Since there is a balanced distribution between positive and negative reviews, there is no need for sampling. For starters, we applied the optimal word embedding model catered to each algorithm on the whole dataset. The computed word embedding values will then be split into training and test dataset by ratio of 8:2, with random seed (40) to ensure the same training and testing set are used for every modelling.

The sentiment analysis models used were: Logistic Regression (LR), Multinomial Naive Bayes (NB) and Random Forest (RF). To improve our models, we applied the following:

- Grid Search (except for NB and RF): Hyperparameter tuning to determine the optimal hyperparameter for each model
- 5-fold Cross Validation (except for RF): Split the training dataset into 5 groups of sub-datasets. The model will train on 4 groups of sub-datasets and test the model on the remaining group of sub-dataset and this process is repeated 5 times with each different subgroup of dataset as testing data. This is to estimate how well the model can perform in general on data not used for training the model and to avoid overfitting the model.

6.2.1 Logistic Regression

Logistic Regression measures the relationship between the categorical dependent variable (positive or negative target labels) and one or more independent variables (TF-IDF Bigram values of text) by estimating probabilities using a logistic/sigmoid function. We use 5-fold cross validation with GridSearchCV which helps to find the best hyperparameter for Logistic Regression and tells the general performance of model on unseen data. Figure 16 below shows the best hyperparameters and cross validation score:

```
Best cross-validation score: 0.95
Best parameters: {'C': 1}
Best estimator: LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
    intercept_scaling=1, l1_ratio=None, max_iter=100,
    multi_class='warn', n_jobs=None, penalty='l2',
    random_state=None, solver='warn', tol=0.0001, verbose=0,
    warm_start=False)
```

Figure 16: Logistic Regression Cross Validation Score and Optimal Hyperparameters

The optimal hyperparameters were then used to train the model and test on unseen data. The top 25 positive and negative words classified by the model during training are indicated in Figure 17 below together with their coefficient estimates:

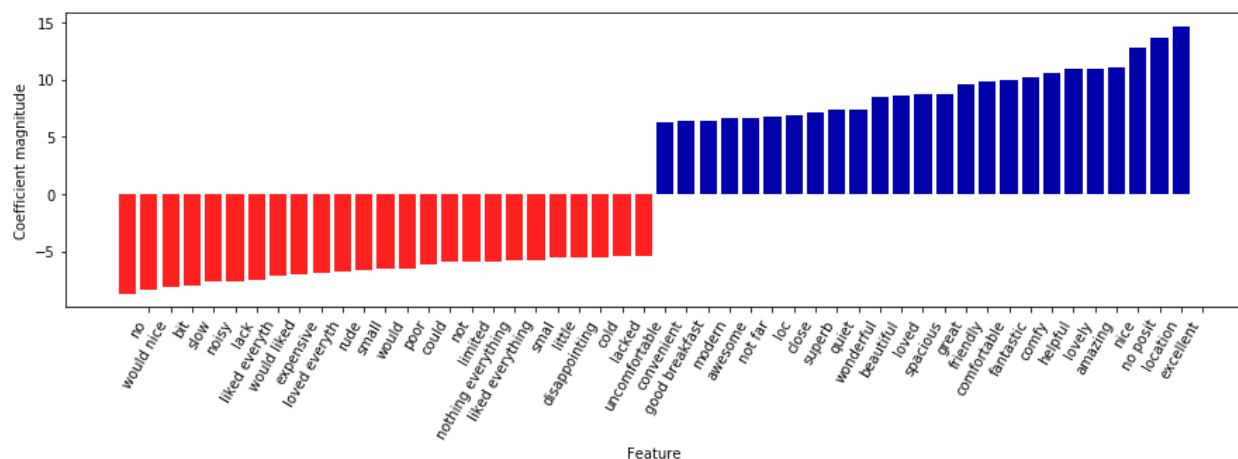


Figure 17: Top 25 Word Features for Positive and Negative Sentiment

Most of the adjectives which have positive connotations such as “wonderful”, “beautiful”, “excellent” and “spacious” were classified as positive sentiments whereas adjectives which have negative connotations such as “small”, “rude” and “poor” were classified as negative sentiments, which is the expected result. However, some words like “would” and “could” were classified as negative words, even though in normal context, they are neutral words. At the same time, words such as “liked everything” and “loved everything” were also classified as negative words, even though they have positive connotations in normal basis. This means that some words classified in the model are highly dependent on context, indicating that these words are often used in

negative reviews. This means users may use these words when suggesting improvement or what they hope to gain.

After training, we then tested the model on test dataset, which achieved an accuracy of 95.1% and AUC 0.99, which means that Logistic Regression has excellent performance in classifying positive and negative reviews. It is also noted that the training accuracy (96.1%) and test accuracy were very close, which means the model is not overfitting. Figures 18 and 19 below show the detailed output of prediction from the model on test dataset:

	actual_label	prediction	abstract
0	pos	pos	busy popular hotel relaxed quiet good night s...
1	neg	neg	the beds poor two single b
2	neg	neg	the noise think walls thin could heard bathro...
4	pos	pos	staff helpful friendly location perfect purpo...
5	neg	neg	very hard bed difficult find hotel walk left ...

Figure 18: Correct Prediction

	actual_label	prediction	abstract
3	neg	pos	toast breakfast plenty choos
12	neg	pos	so quiet bor
43	neg	pos	the waitress breakfast unfriendly accommodati...
46	pos	neg	space area quite valu
50	neg	pos	it old furniture good breakfast

Figure 19: Incorrect Prediction

For incorrect prediction output, we noticed that if sentences contain words such as “breakfast” and “quiet”, they are predicted as positive sentiment whereas their actual labels are negative sentiment. This is because these words often appear in reviews with positive labels and therefore during training, these words are assigned positive sentiments (as indicated in Figure 19). Therefore, this resulted in wrong prediction.

6.2.2 Multinomial Naive Bayes

Naive Bayes is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. A Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Here, multinomial Naive Bayes is used as it is most appropriate for text features that represents word count or count rates and assume the features follow simple multinomial distribution. In this context, Bayes’ Theorem measures the probability of a sentence is positive or negative label given the words is equal to the probability of the sentence is positive or negative label (hypothesis) multiplied by the probability of the words (evidence) given the hypothesis, then divided by the probability of the evidence. For example, whether the review “busy popular hotel” will be given label positive or negative sentiment will be based on the calculation below:

$$\begin{aligned}
 &Pr(\text{positive}) * Pr(\text{"busy"}|\text{positive}) * Pr(\text{"popular"}|\text{positive}) * Pr(\text{"hotel"}|\text{positive}) \\
 &\quad \text{vs} \\
 &Pr(\text{negative}) * Pr(\text{"busy"}|\text{negative}) * Pr(\text{"popular"}|\text{negative}) * Pr(\text{"hotel"}|\text{negative})
 \end{aligned}$$

If the probability of the review is positive sentiment given the presence of words is greater than the probability of the review is negative, then Naive Bayes will return positive label and vice

versa. Since we are using TF-IDF Trigram as word embedding for Naive Bayes, the words will be replaced by the TF-IDF values of words with 1-3 consecutive words.

Before training, we also applied 5-fold cross validation. However, there is no need to tune hyperparameters since there are no hyperparameters to tune for Naive Bayes classifier.

```
Best cross-validation score: 0.94
Best parameters: {'alpha': 1}
Best estimator: MultinomialNB(alpha=1, class_prior=None, fit_prior=True)
```

Figure 20: Multinomial NB Cross Validation Score and Default Hyperparameters

After training, we then tested the model on test dataset, which achieved an accuracy of 94.3% and AUC 0.99, which means that Naive Bayes also has excellent performance in classifying positive and negative reviews. This could be because Naive Bayes tend to work better on high dimensional data - it is much less likely for any two text features to be found close together and text feature cluster tend to be more separated in high dimensions. It is also noted that the training accuracy (94.5%) and test accuracy are very close, which means the model is not overfitting. Figures 21 and 22 below show the detailed output of prediction from the model on test dataset:

	actual_label	prediction	abstract
0	pos	pos	busy popular hotel relaxed quiet good night s...
1	neg	neg	the beds poor two single b
2	neg	neg	the noise think walls thin could heard bathro...
4	pos	pos	staff helpful friendly location perfect purpo...
5	neg	neg	very hard bed difficult find hotel walk left ...

Figure 21: Correct Prediction

	actual_label	prediction	abstract
3	neg	pos	toast breakfast plenty choos
12	neg	pos	so quiet bor
17	pos	neg	like room decor
43	neg	pos	the waitress breakfast unfriendly accommodati...
46	pos	neg	space area quite valu

Figure 22: Incorrect Prediction

Although the incorrect prediction output is almost similar to that of logistic regression, it has slightly more incorrect prediction. It could be due to some words (more than 3-grams) are conditionally dependent on each other, especially since we have large feature space, therefore resulting in an incorrect prediction.

6.2.3 Random Forest

Random Forest is a supervised learning algorithm. It is also the most flexible and easy to use algorithm. It creates decision trees on randomly selected sample reviews, gets prediction from each tree and selects the best solution by means of voting. The more decision trees, the more robust the model. It also provides a pretty good indicator of the feature importance. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio and Gini index for each attribute. Each tree votes and the most popular prediction result is chosen as the final result.

As hyperparameter tuning using grid search and cross fold validation on Random Forest model are computationally expensive and have long runtime, they are thus not applied on this model. As a result, the training accuracy (0.983) is higher than test accuracy (0.934) which means that the model has overfitted. Figures 23 and 24 below show detailed output of prediction:

	actual_label	prediction	abstract
0	pos	pos	busy popular hotel relaxed quiet good night s...
1	neg	neg	the beds poor two single b
2	neg	neg	the noise think walls thin could heard bathro...
3	neg	neg	toast breakfast plenty choos
4	pos	pos	staff helpful friendly location perfect purpo...

Figure 23: Correct Prediction

	actual_label	prediction	abstract
12	neg	pos	so quiet bor
43	neg	pos	the waitress breakfast unfriendly accommodati...
46	pos	neg	space area quite valu
50	neg	pos	it old furniture good breakfast
72	neg	pos	the concierge crew extremely help

Figure 24: Incorrect Prediction

6.2.4 Comparison of Results

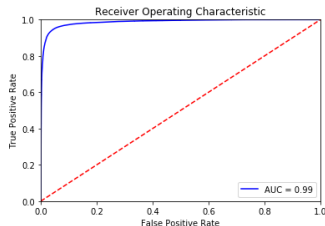
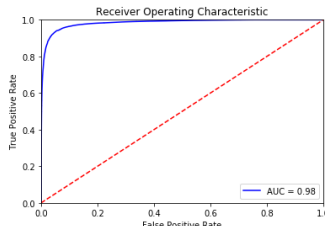
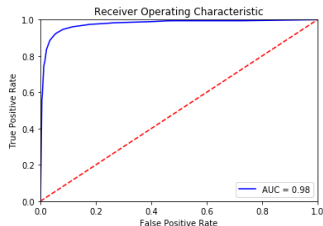
Stemming	Logistic Regression	Multinomial NB	Random Forest																											
Training Accuracy	0.961	0.945	0.983																											
Test Accuracy	0.951	0.943	0.934																											
Confusion Matrix	<table><tr><td></td><td>Neg</td><td>Pos</td></tr><tr><td>Neg</td><td>98267</td><td>4477</td></tr><tr><td>Pos</td><td>5612</td><td>97729</td></tr></table>		Neg	Pos	Neg	98267	4477	Pos	5612	97729	<table><tr><td></td><td>Neg</td><td>Pos</td></tr><tr><td>Neg</td><td>97122</td><td>5622</td></tr><tr><td>Pos</td><td>6170</td><td>97171</td></tr></table>		Neg	Pos	Neg	97122	5622	Pos	6170	97171	<table><tr><td></td><td>Neg</td><td>Pos</td></tr><tr><td>Neg</td><td>96872</td><td>5872</td></tr><tr><td>Pos</td><td>7705</td><td>95636</td></tr></table>		Neg	Pos	Neg	96872	5872	Pos	7705	95636
	Neg	Pos																												
Neg	98267	4477																												
Pos	5612	97729																												
	Neg	Pos																												
Neg	97122	5622																												
Pos	6170	97171																												
	Neg	Pos																												
Neg	96872	5872																												
Pos	7705	95636																												
ROC Curve & AUC	 <p>AUC=0.99</p>	 <p>AUC=0.98</p>	 <p>AUC=0.98</p>																											
Precision	0.956	0.945	0.942																											
Recall	0.946	0.940	0.925																											
F1-score	0.951	0.943	0.934																											

Figure 25: Summary of Results (Stemming)

Generally, the test accuracy of all three models were fairly high (>93%), with quite balanced precision and recall scores. AUC areas were also very high (0.98). The good performance of these classification models could be attributed to our dataset quality being high without much corrupted data samples (e.g. spelling errors, incoherent sentence, vulgarity) even though the dataset is very large. Among the three models, Logistic Regression had the best performance followed by Naive Bayes and Random Forest. In terms of runtime, Naive Bayes was the fastest followed by Logistic Regression and Random Forest. Additionally, we also considered Support Vector Machine (SVM) as one of the models. However, its computation time was too long (>3 days) for over 1 million reviews. Therefore, it was not included in our final model list.

6.3 Further Extension

To further improve our models, we decided to explore different cleaning techniques to see if there is any difference in accuracy to our models.

6.3.1 Lemmatization vs Stemming

Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the language. In comparison, lemmatization reduces the inflected words properly ensuring that the root word belongs to the language. In this project, stemming is conducted using PorterStemmer function from NLTK package while lemmatization is conducted using lemma function from spaCy package. For spaCy package, important part of speech tags (Nouns, Verbs, Adjectives and Adverbs) were specified to keep the particular word form in reviews.

	Stemming	Lemmatization
0	only park outside hotel beauti	park hotel beautiful angry make post available...
1	no real complaints hotel great great location...	real complaint hotel great great location surr...
2	location good staff ok it cute hotel breakfas...	location good staff cute hotel breakfast range...
3	great location nice surroundings bar restaura...	great location nice surrounding bar restaurant...
4	amazing location building romantic set	amazing location building romantic setting boo...

Figure 26: Cleaned Text after Stemming and Lemmatization

As a result, the reviews after stemming and lemmatization are slightly different such that words like “only”, “outside” and “ok” were removed after lemmatization while they were still kept under stemming. The difference in output is that a stemmer operates on a single word without knowledge of the context, and therefore cannot discriminate between words which have different meanings depending on the part of speech.

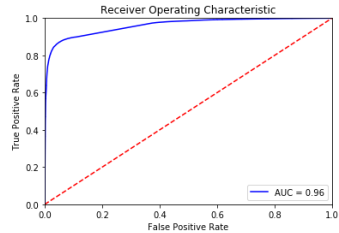
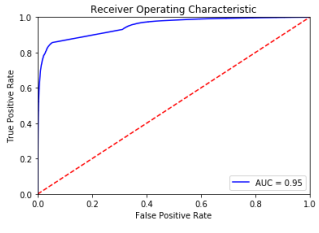
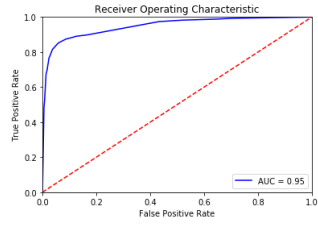
Lemmatization	Logistic Regression	Multinomial NB	Random Forest
Training Accuracy	0.921	0.905	0.945
Test Accuracy	0.910	0.902	0.896
Confusion matrix	Neg Pos Neg 97745 4999 Pos 13515 89826	Neg Pos Neg 97295 5449 Pos 14846 88495	Neg Pos Neg 96472 6272 Pos 15115 88226
ROC Curve & AUC	 <p>AUC=0.96</p>	 <p>AUC=0.95</p>	 <p>AUC=0.95</p>
Precision	0.914	0.906	0.933
Recall	0.910	0.902	0.853
F1-score	0.910	0.902	0.892

Figure 27: Summary of Results (Lemmatization)

The accuracy, precision, recall and F1-score of models after lemmatization have decreased by approximately 4%. This could be due to some words removed after lemmatization being important in determining the sentiment of reviews, therefore resulting in difference of TF-IDF values and thus difference of model performance.

6.3.2 Convolutional Neural Network (CNN)

While the algorithmic approach such as Logistic Regression, Naive Bayes and Random Forest are easy to implement and efficient, they learn from what is in a class, but not what is not. The learning of patterns of what does not belong to the positive class or negative class is often important. Besides, if applied to real world data, more often than not, the data may not have balanced distribution between different target labels. Classes with disproportionately large training sets can create distorted classification scores, forcing the algorithm to adjust scores relative to class size, which is not ideal. Therefore, we decided to test the performance of CNN on hotel reviews.

Before text is passed into CNN, it must be converted into vectors or sequence as input to CNN. Each word in review is assigned an integer using `Tokenizer.text_to_sequences()` and that integer is placed in a list. As all the sentences must have the same input shape, we pad the sentences. For example, if we have a sentence “busy popular hotel quiet good relaxed”, each word is assigned a number. We suppose busy=1, popular=2, hotel=3, quiet=4, good=5 and relaxed=6. After `texts_to_sequences` is called, our sentence will look like [1, 2, 3, 4, 5, 6]. Since the longest string in the review is 1863 and we want to capture as many words in the review as possible to prevent loss of information, we set our MAX_SEQUENCE_LENGTH=1000. After padding our sentence will be [0, 0, 0,, 1, 2, 3, 4, 5, 6]. The total tokens for each padded sentence will be 1000.

Once the word embedding matrix is computed, we split it into training and test dataset by ratio of 8:2. The embeddings matrix is passed to embedding_layer with arguments as follows: total size of vocab=80133 (the number of unique words+1), dimensions=50, length of input vector=1000(MAX_SEQUENCE_LENGTH). The output from the Embedding layer will be 1000 vectors of 50 dimensions each, one for each word. We flatten this to a one 50,000 element vector to pass on to the Dense output layer. Convolution1D layer is added to learn filters with GlobalMaxPooling1D(). Since the output target is binary, the loss function is binary cross entropy and activation function used in the output layer is sigmoid function. A dropout layer and then final dense layer is applied.

```
Build model...
Train on 551088 samples, validate on 137772 samples
Epoch 1/2
551088/551088 [=====] - 32018s 58ms/step - loss: 0.2179 - acc: 0.9141 - val_loss: 0.1758 - val_acc: 0.9341
Epoch 2/2
551088/551088 [=====] - 22919s 42ms/step - loss: 0.1751 - acc: 0.9357 - val_loss: 0.1655 - val_acc: 0.9387
```

Figure 28: 1D CNN Training and Validation Accuracy and Loss

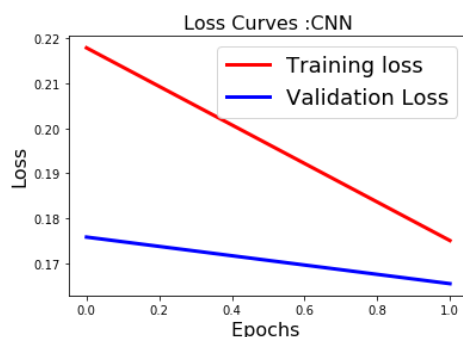


Figure 29: Loss over Epochs

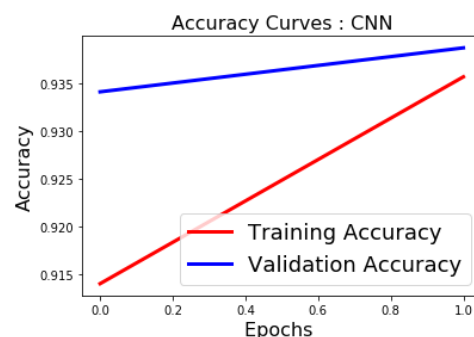


Figure 30: Accuracy over Epochs

The accuracy of CNN on test dataset is 0.939, which is quite high and comparable to Logistic Regression, Naive Bayes and Random Forest. However, the epochs used is only 2. Epochs is

the number of times a learning algorithm sees the complete dataset. Since the number of epochs used is very low, this could mean that there is not enough training for neural network. From Figures 29 and 30, it is observed that the loss decreased and accuracy increased as the number of epochs increased. However, they have yet to converge, which means epochs should be further increased.

Unfortunately, the dataset is too large such that it is computationally expensive to train neural network on large number of epochs. Each epoch took 8 hours to train. Due to time constraints and insufficient memory allocation in Jupyter Notebook, epoch used in this model was reduced to 2.

7. Topic Modelling

To discover what customers talk about in their reviews, we performed topic modelling on positive and negative reviews separately to distinguish between areas customers are satisfied and dissatisfied with.

7.1 Define Model Parameters

We used a Latent Dirichlet Allocation (LDA) model to carry out topic modelling. We chose LDA over other models like Latent Semantic Analysis (LSA) because LDA adds an additional probabilistic distribution layer which prevents overfitting and produces better results.

Besides basic model parameters like *corpus*, *id2word* and *num_topics*, we also specified several others we found important, namely *random_state*, *chunksize* and *passes*. *random_state* is the seed used by the random number generator. We set it to 100 for model reproducibility. *chunksize* is the number of documents processed in each training chunk while *passes* is the number of passes through the corpus during training. We set them to 100 and 10 respectively to balance between model quality and training speed.

7.2 Find Optimum Number of Topics

We determined the optimum number of topics based on coherence value, which measures the relative distance between words in a topic. The higher the coherence value, the smaller the distance between words and hence, the more similar they are. As such, we built several LDA models, each with different number of topics, and picked the one that resulted in the highest coherence value. We restricted the number of topics to range from 2 to 10. We did not want the number of topics to be less than 2 as that would be meaningless and insignificant for analysis. We also did not consider number of topics that are more than 10 as having too many topics might trigger overlaps in keywords, polluting results.

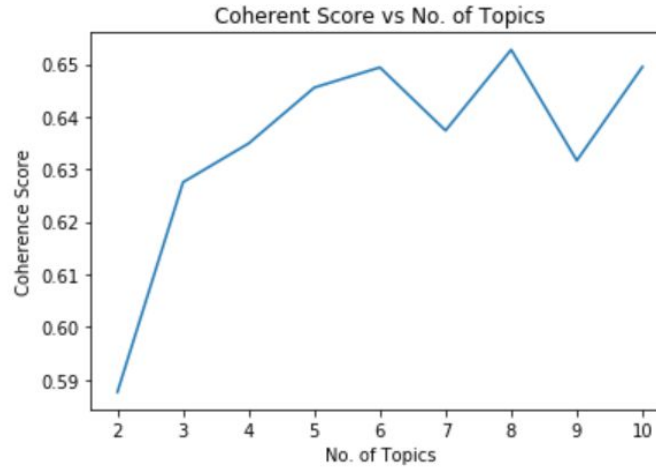


Figure 31: Number of Topics Graph for Positive Reviews

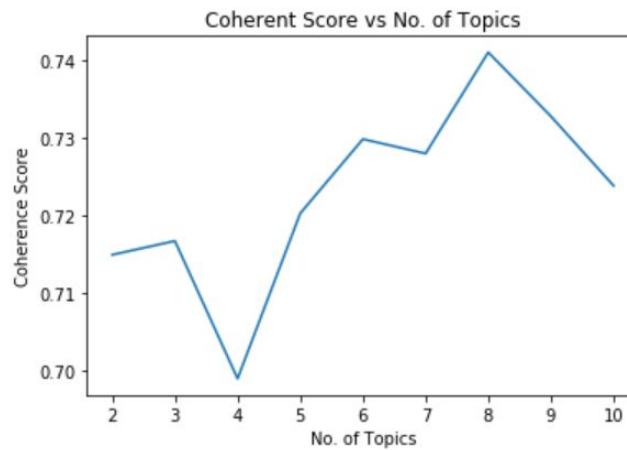


Figure 32: Number of Topics Graph for Negative Reviews

After trying and computing the coherence values corresponding to 9 different LDA models for both positive and negative reviews, we plotted coherence score versus number of topics graphs for easier visualisation. From Figures 31 and 32 above, we can observe that the optimum number of topics is 8 for both positive and negative reviews. Hence, we selected the two models corresponding to this number of topics.

7.3 Visualise Topics and Related Keywords

We generated word clouds for the top 10 words in each topic. From Figure 33 below, we can infer that the topics for positive reviews are: hotel environment, facility, general vibe, room quality, staff service, location, surroundings and customer opinion. These are areas most customers are satisfied with.

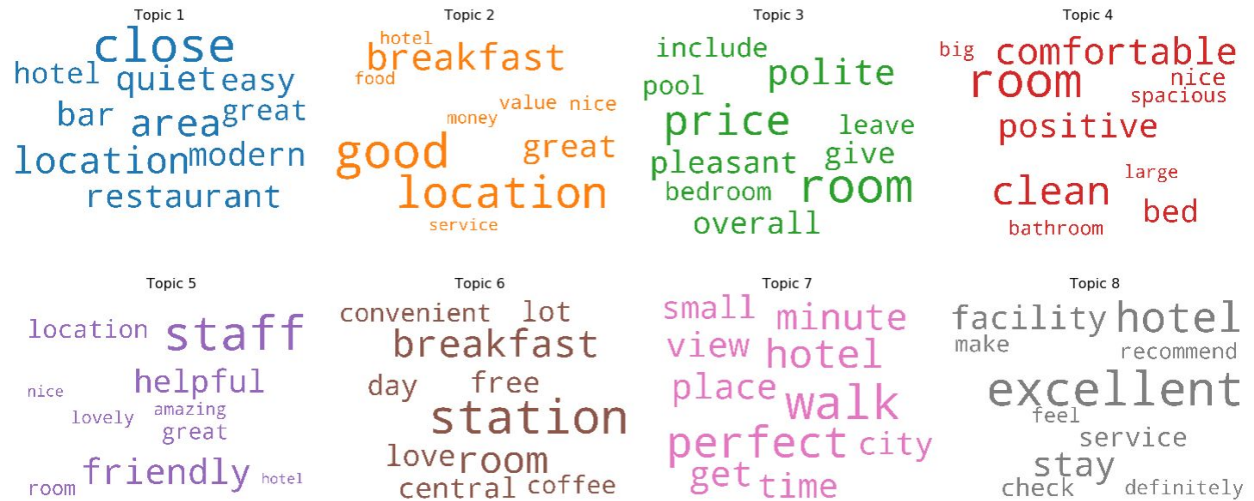


Figure 33: Word Clouds for Positive Reviews

On the other hand, from Figure 34 below, we can infer that the topics for negative reviews are: customer opinion, hotel location, surroundings, staff service, facility, room quality, payment and room cleanliness. These are areas many customers found unsatisfactory and where hotels can improve in. However, we noticed that certain words with positive connotations, such as “good” or “clean”, appeared in these word clouds as well. This could be due to sarcasm in the language used by customers when leaving reviews.



Figure 34: Word Clouds for Negative Reviews

8. Future Improvements

Due to time constraints and computational power, we were unable to play around with the data we have at hand extensively. If given more time, below are some areas we would like to explore as well:

8.1 Further Sentiment Analysis Model Exploration

Given that our dataset is high dimensional with many number of features, we can consider other models such as XGBoost or MART which beats the curse of dimensionality through adaptive adjustment of neighbourhoods. They are able to perform automatic feature selection and capture high-order interactions without breaking down.

As for word embedding techniques such as TF-IDF, it does not capture the semantic of words (meaning of words). Besides, while the use of n-grams takes into account the order of words in a sentence, it also has the risk of adding noise into the model with increased number of text features which could be redundant. A possible alternative to word embedding could be Word2Vec. There are two models in this class: CBOW and Skip-grams and these are a set of neural network models that have the aim to represent words in the vector space. These models are highly efficient and performant in understanding the context and relation between words. Similar words are placed close together in the vector space while dissimilar words are placed wide apart.

Given more time, hyperparameter tuning and 5 fold cross validation can be used during Random Forest algorithm to reduce overfitting. To increase time computation efficiency, we can take a sample of dataset for training all algorithms instead of taking the whole dataset which is computationally expensive with 1 million reviews. This could help Random Forest and most importantly neural network to be trained faster.

8.2 Grouped Topic Modelling

Currently, we only explored topics for positive and negative reviews on the whole. Instead, we could perform topic modelling over time to analyse if what customers talked about have changed over the years. Besides that, we could also group reviews by country or nationality to analyse if customers from different countries or of different nationalities talk about different topics in their reviews.

8.3 Single Sentiment Analysis and Topic Modelling Pipeline

We currently perform sentiment analysis and topic modelling in isolation. We could look into combining sentiment analysis and topic modelling into a single pipeline, where we first identify topics across all reviews, then analyse sentiments of reviews under each topic and assign sentiment scores before classifying them as positive or negative.

9. Possible Extensions

Below includes some ideas we had in mind but were unable to implement due to limit of knowledge:

9.1 Standardised Rating Generator

As mentioned in Section 1, ratings are subjective and may even be biased. Take the two reviews for Renaissance Downtown Hotel in Dubai (Figure 35) as examples. Both of them mentioned windows being dirty as the only downside of the hotel but the two users gave differing ratings of 9.6 and 10.



Figure 35: Reviews with Subjective Ratings

As such, we wanted to create a standardised rating generator, where we assign ratings based on review sentiments to be more standardised and unbiased.

9.2 Hotel Recommender

We also thought of building a hotel recommender system, where we could recommend hotels to users based on three different premises: One, to analyse hotel descriptions and identify and recommend those that might be of interest to users. Two, to recommend hotels based on those with the largest cosine similarity to past hotels booked or viewed by users. Three, to recommend based on user-specified criteria like room cleanliness or spaciousness.

10. Conclusion

Given the evaluation of the models for sentiment analysis, it is deduced Logistic Regression has the best performance in terms of accuracy, precision, recall and F1-score. In comparison, Naive Bayes also has relatively good performance and is less likely to overfit the dataset. Despite having high accuracy, Logistic Regression and Random Forest can easily suffer from overfitting if there are too many text features and without hyperparameter tuning. Also, the implementation of CNN for sentiment analysis will be feasible if the dataset is reduced.

Thus, the large number of word features generated should be reduced with proper feature selection techniques to avoid overfitting. The features that will be included in the model will either be determined by the user or by a user-defined threshold.

In tandem to the business use case, Logistic Regression with reduced number of text features will be the easiest to implement given its computational efficiency and this will be useful in reducing the time spent on the classification of reviews process when the number of reviews that hotels receive scales to larger numbers.

As for topic modelling, the topics for both positive and negative reviews were quite distinct with few overlapping words. This makes it useful for hotels to implement to figure out areas that the customers are satisfied or dissatisfied with, allowing them to improve their services accordingly. However, hotels should be alert to the use of sarcasm in reviews left by customers, which may cause reviews to be wrongly classified and thus, words to be grouped under wrong topics.

11. References

Dataset:

Liu, J. (2017, August 21). 515K Hotel Reviews Data in Europe. Retrieved from <https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe>.

Preprocessing:

Smetanin, S. (2018, September 1). Sentiment Analysis of Tweets using Multinomial Naive Bayes. Retrieved from <https://towardsdatascience.com/sentiment-analysis-of-tweets-using-multinomial-naive-bayes-1009ed24276b>.

Sentiment Analysis:

モルラ, モハマド M. (2018, October 12). Machine Learning: Sentiment analysis of movie reviews using LogisticRegression. Retrieved from <https://itnext.io/machine-learning-sentiment-analysis-of-movie-reviews-using-logisticregression-62e9622b4532>.

Agarwal, R. (2019, July 21). NLP Learning Series: Part 3 - Attention, CNN and what not for Text Classification. Retrieved from <https://towardsdatascience.com/nlp-learning-series-part-3-attention-cnn-and-what-not-for-text-classification-4313930ed566>.

Topic Modelling:

Topic Modeling in Python with Gensim. (2018, December 4). Retrieved from <https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/>.

Word Clouds of Top N Keywords in Each Topic:

Topic modeling visualization - How to present results of LDA model?: ML . (2018, December 4). Retrieved from <https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/#9.-Word-Clouds-of-Top-N-Key-words-in-Each-Topic>.