CMM706 - Text Analytics | Study Report

Kaneel Dias
IIT ID: 20242001
RGU ID: 2506673

Part A Summary of overall findings

The overall code used for all tasks in this assessment can be found in the GitHub repository below:
https://github.com/kaneeldias/sri-lanka-hotel-reviews/

A.1 Data collection

The reviews were collected using a Python script utilizing the Trip Advisor API. If a common search term like "Sri Lanka" were used to retrieve the reviews, most reviews would come from the most popular hotels from the most popular regions. In order to ensure diversity in the reviews, first a list of the top 100 cities in Sri Lanka was generated using Google's Gemini along with their respective latitudes and longitudes.

For each city in this list, the nearby search endpoint of the Trip Advisor API was used to collect a list of hotels. Then for each hotel, the location reviews were used to fetch a set of reviews up to a maximum of 25 reviews per hotel.

All the reviews were processed to collect only the most relevant fields, and then saved to a CSV file. The final CSV file contained 7,367 reviews.

A1.1 Challenges in collecting data

Inconsistencies in the Trip Advisor API

- The pagination feature was sometimes available, and at other times not
- Different queries sometimes returned duplicate reviews
- No. of results returned per API call changes

Limitations of the Trip Advisor API

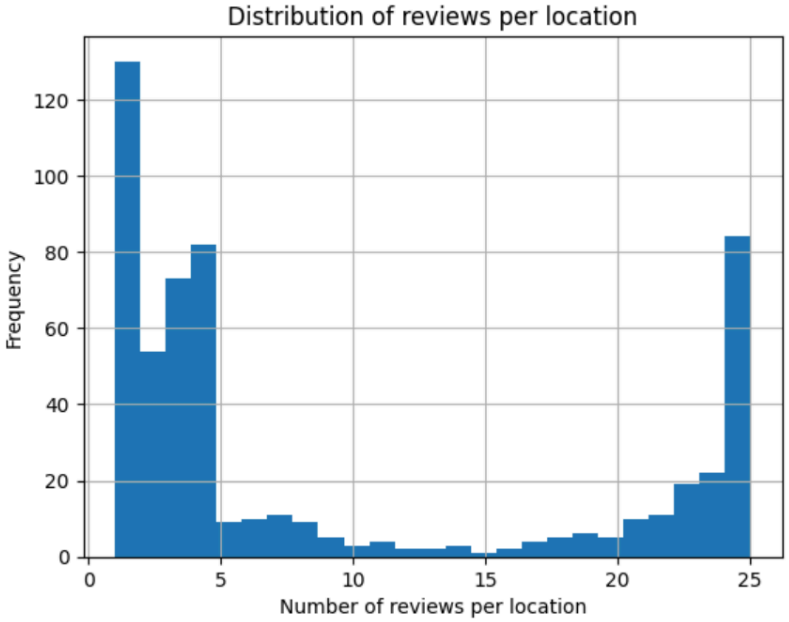
- The API only allowed 5000 API calls per month under the free tier
- Only a limited number of reviews were retrievable per hotel
- Inability to perform generic location based searches

Network issues

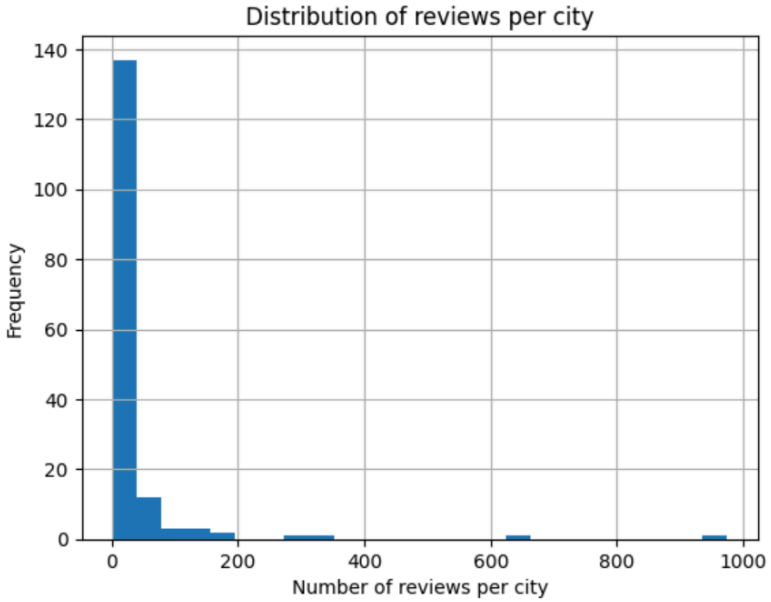
- Due to network issues, the script was interrupted and needed to be re-run, exhausting the free tier API limits and collecting duplicate reviews. Several workarounds were later added to mitigate this.

Imbalanced Dataset

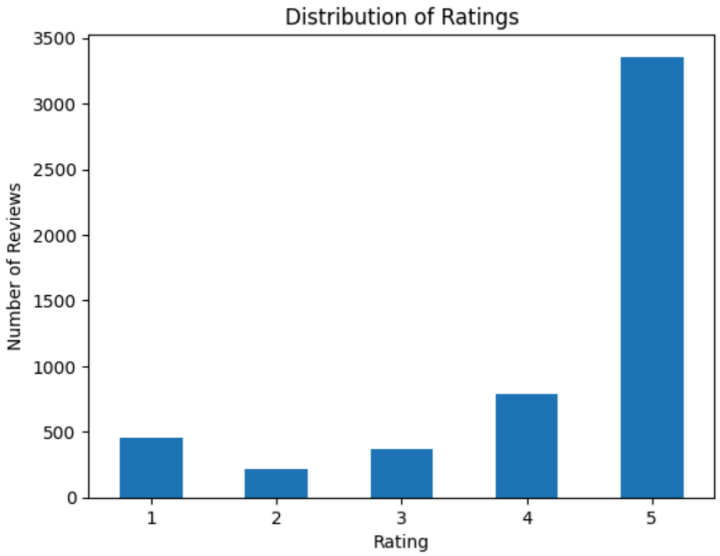
- The dataset obtained was heavily imbalanced, both in terms of the reviews per location and the rating (positive reviews were much higher than negative reviews)



Imbalance in reviews per location



Imbalance in reviews per city



Imbalance in ratings

A.2 Cleaning the data

The following steps were taken to clean the data obtained from the API

- Removal of duplicate reviews brought the total number of reviews from 7,367 down to 5,186.
- All reviews were converted to lowercase
- All non-alphanumeric characters were removed
- Leading and trailing whitespaces were tripped
- Stopwords were removed
- The review title and text were concatenated to form one unified review string

A.3 Establishing ground truth

Three different sentiment classifiers were used to determine the sentiment polarity of each review.

- **TextBlob**
  - Provides a sentiment value between -1 and +1
  - This was converted to a binary value of 1 (positive) or 0 (negative) using a threshold of 0
- **Valence Aware Dictionary and sEntiment Reasoner (VADER)**
  - Provides four values: negative, neutral, positive and compound.
  - The compound value is a normalized score that ranges from -1 to +1
  - This compound value was converted to a binary value of 1 (positive) or 0 (negative) using a threshold of 0
- **Transformer**
  - A sentiment analysis transformer using the `distilbert-base-uncased-finetuned-sst-2-english` model was used.
  - This provides two outputs: score and label, where score is the confidence score of the sentiment label and label is either 'POSITIVE' or 'NEGATIVE'.
  - We determine the sentiment polarity using the label only/

In all three cases, commonly available Python modules were used which can be found in the import section here.

The ground truth was obtained using a majority voting system where if at least 2 models returned the same answer, that value was used.

A.4 Feature extraction

All reviews were converted to the following representations

- Bag of Words (sparse vector representation)
- TF-IDF (sparse vector representation)
- Word2Vec (dense vector representation)
- Doc2Vec (dense vector representation)

In all four cases, commonly available Python modules were used which can be found in the import section here.

A.5 Text classification

Three different classifiers were used with the above four text representations.

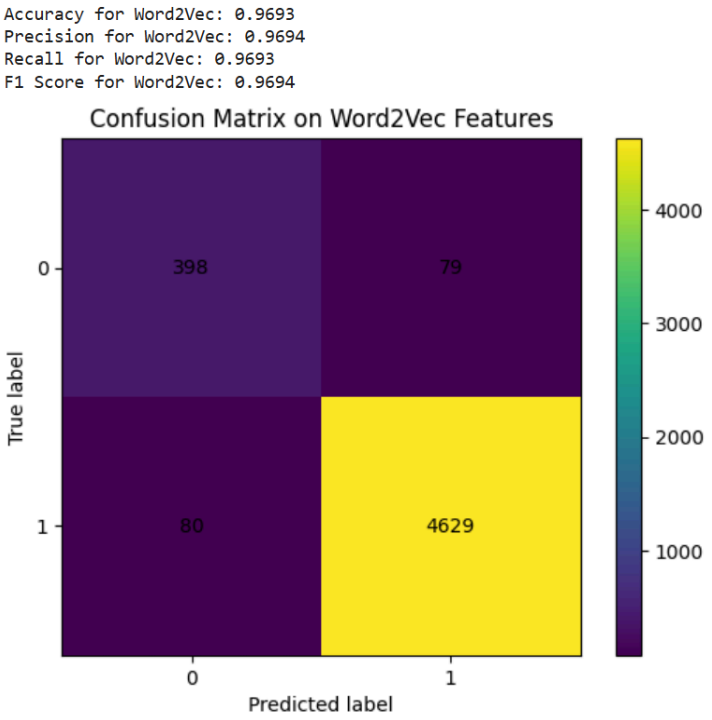
- Naive Bayes
- Support Vector Machine (SVM)
- Random Forest

The F-1 score was used as the primary metric to measure performance and the respective value for each classifier-representation combination can be found below.

	naive_bayes	svm	random_forest
bow	0.959209	0.961399	0.932673
tfidf	0.931996	0.961679	0.927734
word2vec	0.951042	0.969355	0.962778
doc2vec	0.864625	0.902497	0.876121

F-1 scores of the classifiers

The best performing model was the SVM using Word2Vec.



Performance of the SVM using Word2Vec

A.6 Using pre-trained vectors

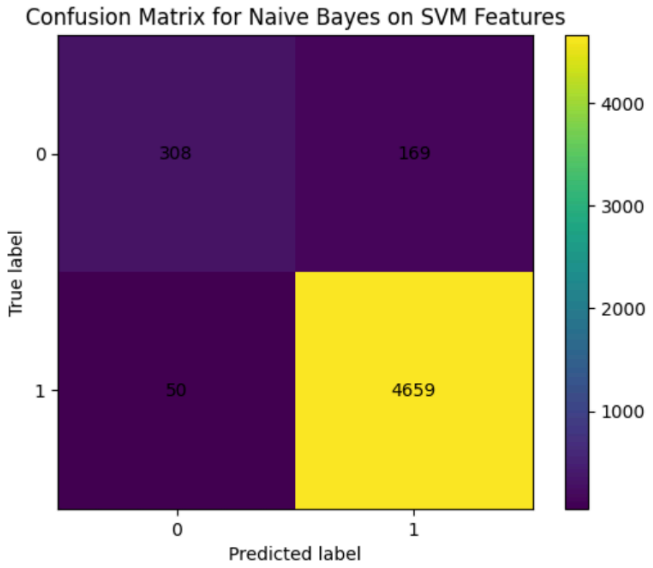
BERT was used as the pre-trained text embedding vector for this task.

A.6.1 Using non-deep learning algorithms

This was used against the same four classifiers from the previous task.

Once again, the SVM classifier shown the best performance

```
Accuracy for SVM: 0.9578
Precision for SVM: 0.9554
Recall for SVM: 0.9578
F1 Score for SVM: 0.9550
```



Performance of the SVM using BERT embeddings

However this implementation's performance did not surpass that of the SVM using Word2Vec.

A.6.2 Using a Neural Network

A neural network was used and its hyperparameters were tuned to obtain the best possible validation accuracy.

```
model = Sequential([
    Input(shape=(X_train.shape[1],)),
    Dense(512, activation='relu'),
    Dropout(0.6),
    Dense(512, activation='relu'),
    Dropout(0.6),
    Dense(2, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Final hyper-parameters used

The final F-1 score obtained was 0.9562, which still did not surpass that of the SVM using Word2Vec.

A.7 Topic modelling, clustering, and building an aspect based sentiment classifier

A.7.1 Topic modelling

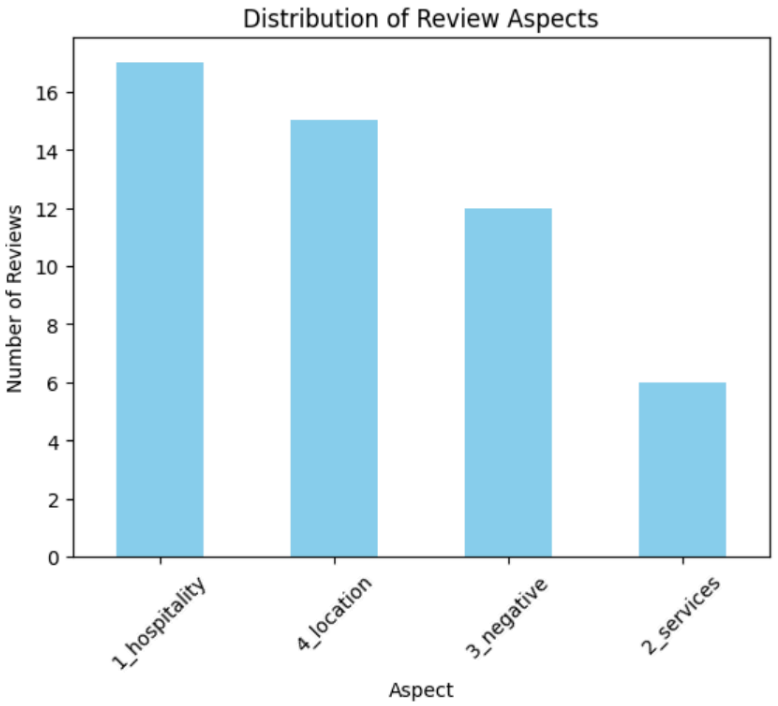
The LDA algorithm was used for topic modelling. After several iterations of trial and error, it was settled upon that 4 would be the ideal number of topics. Using this value, 4 topic themes were identified.

- Topic 1 - Positive feedback on hotel staff and hospitality
- Topic 2 - Positive feedback on hotel amenities and services
- Topic 3 - Negative feedback
- Topic 4 - Positive feedback on hotel location and surrounding (indirect to the hotel itself)

Regardless of the combinations attempted, I was unable to cluster for topics based on different aspects.

A.7.2 Manual labelling

50 random reviews were selected, and manually labelled according to the four topics identified above.

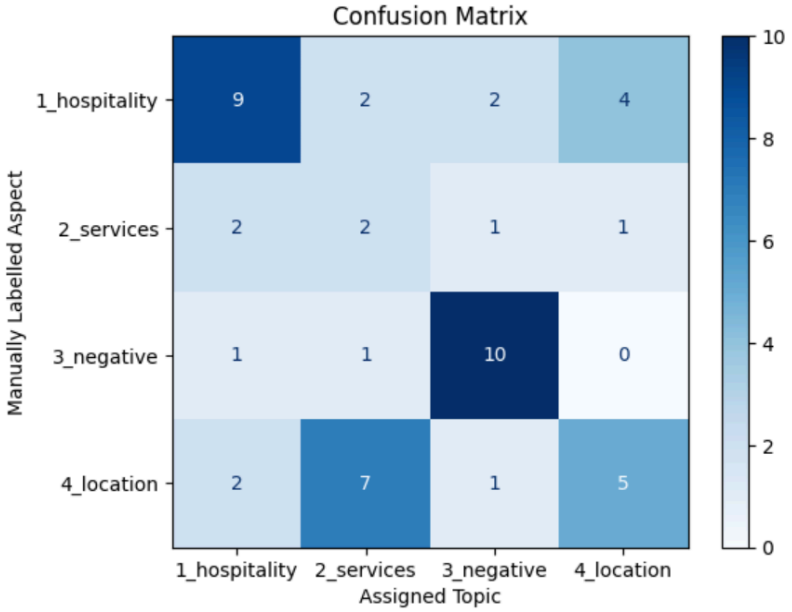


Distribution of manual labels

The manual labelling process itself was very difficult with significant ambiguity where some reviews could be categorized into more than one topic.

A.7.3 Evaluation of the clustering

These manually labelled reviews were applied on the LDA model trained.



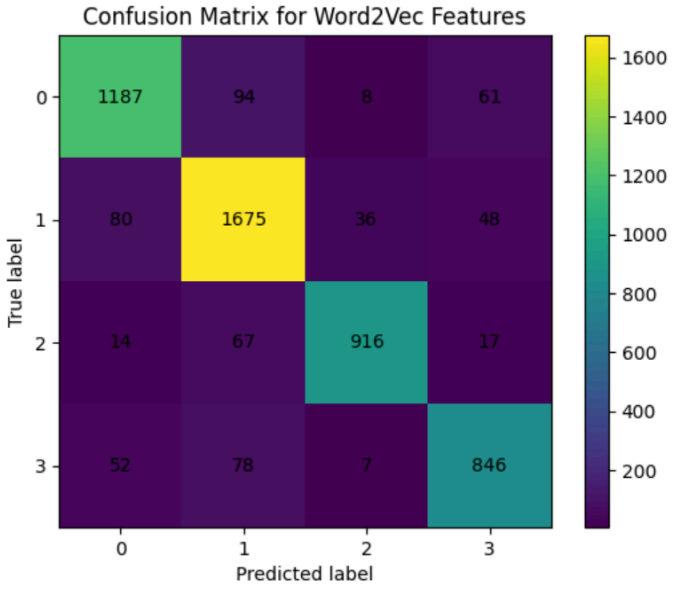
Confusion matrix for the manual labels vs. assigned topics using LDA

The performance wasn't great, with an accuracy of only 0.5200, Rand Index of 0.6988 and Jaccard Coefficient of 0.3523.

A.7.4 Implementing an aspect-based sentiment classifier

The LDA model we trained earlier was used to apply cluster labels to all 5000+ reviews.

Then our best performing classification model (SVM with Word2Vec) was used to classify the reviews according to these labels.




```
0.8917355182703203
```

Performance of the aspect based sentiment classifier using SVM and Word2Vec

A good performance was observed, with an F-1 score of 0.8917.

A.8 Final recommendations

- Ensure diversity in the dataset by collecting more negative and lower-rated reviews
- Collect data from other sources as well (Booking.com, Agoda, Google reviews, Expedia etc.)
- Use the already included aspect-based rating in TripAdvisor for the aspect based sentiment classifier.
- Use of RNNs to preserve meaning in the order of words.

Part B on next page

Part B Learning process

During the completion of this assessment, I came across several new concepts which I had not been exposed to before. Some of which include:

- The Latent Dirichlet Allocation (LDA) algorithm used for topic modelling
- The Word2Vec and Doc2Vec algorithms for creating text representations
- How n-grams can be used identify common themes in reviews
- Using sentiment classifiers such as TextBlob and Vader
- Using majority voting as a method for establishing ground truth
- Practical usage of Naive Bayes, SVM and Random Forest

Several online sources were used in order to understand how to implement these using Python

- https://www.analyticsvidhya.com/
- https://towardsdatascience.com/
- https://geeksforgeeks.org/

Most of these sources recommended using the same set of common Python libraries such as the following.

- Natural Language Toolkit (NLTK)
- Gensim
- Huggingface (transformers)
- Scikit-learn
- Keras

For specific implementation details which were not easily discoverable using the official documentation, online support forums such as StackOverflow, and GitHub issues were used.

Although in actual implementation of the tasks, already existing Python libraries were used for all of the above, the key concepts and how they work behind the scenes was important to understand. I mostly watched YouTube videos (example) to get a better understanding of how these algorithms/concepts work.

B.1 Use of generative AI

Generative AI played a role in the completion of this assessment. However I do not believe the integrity of the assessment was compromised due to its usage and its usage was limited to specific scenarios.

- Google's Gemini was used to compile a list of the top 100 cities in Sri Lanka with their latitudes and longitudes for the data collection process.
- Most of this project was written on the Jetbrains PyCharm IDE, with the GitHub Copilot plugin enabled. This plugin provided code completion support.
- GitHub Copilot was used to refine the documentation/explanations on the .ipynb report.
- Google's Gemini and GitHub Copilot was used to help debug some errors which occurred during development.