Sentiment Analysis and A Look on Aspect Based Sentiment Analysis

Amit Kumar Yadav(UI21CS08), Akash Thappa (UI21CS07)

May 13, 2024

1 Introduction

Sentiment analysis is a computational technique used to determine the sentiment or emotional tone expressed in voice of the customer materials such as reviews and survey responses. By analyzing words, phrases, and even emojis, sentiment analysis algorithms classify the sentiment of text as positive, negative, or neutral. This process involves natural language processing (NLP) techniques and machine learning algorithms trained on labeled datasets. Sentiment analysis finds applications in various fields, including market research, social media monitoring, customer feedback analysis, and brand reputation management. Its insights enable businesses to understand public opinion, gauge customer satisfaction, and make data-driven decisions to enhance products and services.

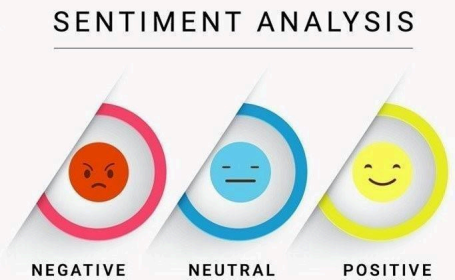

Figure 1: An example of three sentiments-Negative, Neutral, Positive(source)

For example, analyzing customer sentiments and opinions from reviews in the Hotel Industry helps improve the product or service, and make better marketing campaigns. Given the massive amount of textual content, it is intractable to manually digest the opinion information. Therefore, designing an automatic computational frame- work for analyzing opinions hidden behind the unstructured texts is necessary, resulting in the emergence of the research field sentiment analysis and opinion mining

1.1 Types of Sentiment Analysis

1.1.1 Document-level sentiment analysis

Document-level sentiment analysis is a key task in NLP, focusing on determining the overall sentiment expressed in a document, like a review or news article. Unlike aspect-based sentiment analysis, which delves into specific aspects, document-level analysis considers the entire text. Its primary goal is to classify sentiment into predefined categories like positive, negative, or neutral, offering insights into the overall opinion expressed. This has applications in product reviews, brand monitoring, and market research. The process involves several steps like first we remove the noise from the text and make our data clean, then we will extract the features from the data. These features can be the most frequent words or semantic features. With the help of machine learning algorithms. We will classify the sentiment and according to the sentiment we will give every feature its sentiment score. Model performance can be evaluated by the accuracy and F1 score.

1.1.2 Aspect-based sentiment analysis (more)

In sentiment Analysis, understanding the sentiment of the user in the text is very important. The sentiment is dissected in two fundamental components: the target and the sentiment itself. This process is crucial as the aspect and their sentiments in that text will be positive, neutral or negative. ABAS has now done this process in a more detailed way by adding targets in aspect categories and sentiments into opinion terms and sentiment polarity. Aspect categories serve to identify all the unique features or aspects that all the reviewers are mentioned in their review. The Aspect Based Sentiment Analysis pinpoints the terms on which the users are talking about in their review, while sentiment polarity delineates the orientation of these sentiment and categorized it as positive, neutral or negative

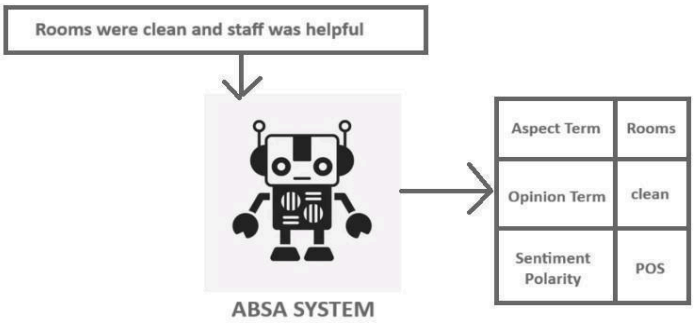

Figure 2: Aspect extraction and aspect polarity(source)

1.1.3 Fine-grained sentiment analysis

Fine grained sentiment analysis is a sophisticated technique to understand the nuances of the emotion that a user expressed in text. In traditional Sentiment analysis we categorized text into border labels such as positive, neutral or negative but in fine-grained sentiment analysis seeks to uncover the emotion of the user like joy, anger, sadness and more. This method helps to understand the emotion of the people's opinions and attitudes. it can distinguish between satisfaction with the product's performance but disappointment with its durability. With the help of the fine-grained analysis understanding the insight of customer emotions, companies can tailor their strategies to better meet customer needs and preferences, ultimately enhancing customer satisfaction and loyalty.

1.1.4 Entity-level sentiment analysis

Entity level sentiment analysis is used to analyze the sentiment towards a specific entity mentioned in the text of the user such products, services, organizations or any other individuals. Its analysis zooms in on the sentiment associated with individual entities within the text. The primary goal of entity level analysis is to evaluate the opinions expressed by users in the text towards different entities.This analysis helps the business to identify the strengths and weaknesses, track changes in customer attitudes towards the different aspects of the business over the time.

1.2 The Importance of Solving the Identified Problem

The customer feedback holds a value that is very useful to the business. With the help of this a business gets to know about the attitude of the customer in different aspects of the business. In the hotel industry the business gets improved on the basis of the customer feedback and customers reacting to the multiple aspects like staff behaviour, room, parking , food and many more.

By focusing on Aspect-Based Sentiment Analysis, we will provide a service which will help to understand the sentiment towards different aspects that will pinpoint the goods and bads in their services. This will provide the useful insights of customers and help to grow the business rapidly.

1.3 Existing Solutions

The review is consistent with the text and nuances the traditional methods use machine learning algorithms trained on label datasets and classify the text into positive, neutral or negative. These methods are effective but only on the text part of the review, they fail to understand the nuances of sentiment that is expressed towards the specific aspect or features. This lack of analysis may affect the decision making of the business.

To overcome this limitation many tools are coporates with the analysis process that involves extracting the features mentioned in the review and then analyzing the review associated with each features. however there are still inaccuracies in extraction process, handling the nuances in the language while dealing with the large datasets.

1.4 Proposed Idea: Aspect-Based Sentiment Analysis on Hotel Reviews

The proposed idea for this project is to develop a Aspect-Based Sentiment Analysis (ABSA) model for hotel reviews that will extract the features from the reviews. This model will use natural language processing (NLP), including deep learning models such as BERT (Bidirectional Encoder Representations from Transformers), to accurately identify aspects mentioned in the reviews and score the sentiment that is associated with those aspects.

Key components of the proposed ABSA model include:

- Aspect Extraction: We first traverse through the whole dataset and extract the 20 most repeating or most used words. Now in these words we filter out the aspects on which we perform our sentiment analysis.
- Sentiment Analysis: We will perform the sentiment analysis on the whole dataset and give sentiment scoring. The scoring is done by comparing the nuances in the review and giving a score to each review.
- Aspect-Level Aggregation: The scoring of each review is now aggregated and we will do the average scoring of each aspect according to the number of reviews that talk about the aspect in their review.
- Visualization and Insights: The result of the aspect based sentiment analysis with the use of matrices. This allows us to gain insight into the strengths and weaknesses of the business on the basis of the aspect that we extracted.

2 Related work/Existing work

There are various methods on this problem that include both traditional and recent methods. These methods use natural language processing techniques and sentiment analysis. There exists a wide range of approaches that use different methods to address various aspects of the problem. Here are some of the techniques.

Lexicon-based Approaches:

In this we use sentiment lexicons sentiment. These methods give a score to the text based on the positive and negative words that users use in their review. the lexicon that we use in VADER in one of our approaches. These are simple and easy to implement but have low accuracy when dealing with nuanced based review.

Machine Learning Models:

It is a popular method that today's world uses in sentiment based analysis, it can recognize complex patterns and make general rules from that pattern. The different techniques used for classification of the sentiment are Support Vector Machines , Random Forests and many more. The BERT and RoBERT are also used to classify the aspect from the user text and then Scoring is done based on that analysis done by these model

Domain-specific Solutions:

These are solutions that are used when you want to perform the sentiment analysis on the particular domain of the industry. In this we will use special models and methods that are used just for that domain. We simply can't use the same aspect in different domains. Like sentiment analysis, healthcare requires a special model that can handle all the terminologies related to the healthcare domain.

Cross-lingual and Multimodal Approaches:

Since people can best express themselves in their mother tongue as they are very comfortable with it. So we need a model that can handle the cross lingual sentiment analysis. The main aim is to capture sentiment in different languages. In multilingual review there is more nuanced data so this will make our model very inaccurate if not implemented in the proper way.

3 Proposed Model

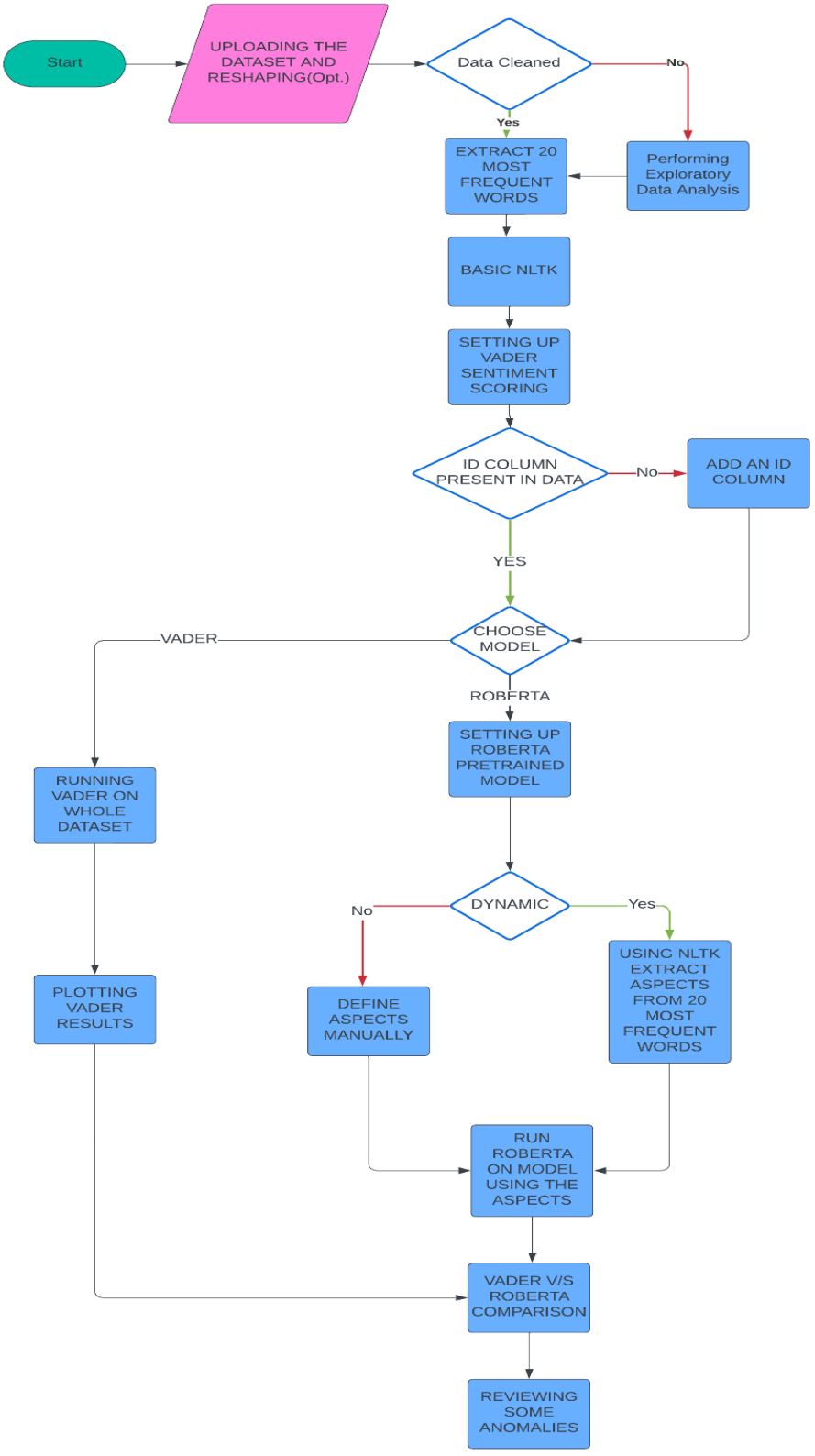

Figure 3: Flowchart

(source)

3.1 IMPORTING THE REQUIRED LIBRARIES

- **Pandas**

  Pandas provides data manipulation and analysis by providing various powerful data structures and methods to make changes on the data like finding missing values,etc . The dataset from google drive is imported into a dataframe which is used in further operations.

- **NumPy**

  NumPy provides numerical computing techniques which include handling multi-dimensional arrays. It is used in generating the values for confusion matrix and other performance metrics.

- **MatplotLib**

  Matplotlib provides visualization by offering a wide range of plotting functions. All the graphs including the confusion matrix are plotted using matplotlib.

- **Seaborn**

  Seaborn is more advanced as compared to matplotlib, it creates attractive statistical graphics by providing customizable themes and color palettes.

- **NLTK**

  NLTK is used to classify the words as different types of English language like Noun,Determinant,Verbs,etc to extract the aspects dynamically.

3.2 UPLOADING THE DATASET

Dataset Link:(LINK)

Google drive is integrated with Colab and then dataset is fetched into a pandas dataframe

Pseudocode:

```
from google.colab import drive
drive.mount('/content/drive')
file_path = '/content/drive/My Drive/tripadvisor_hotel_review.csv'
df = pd.read_csv(file_path)
```

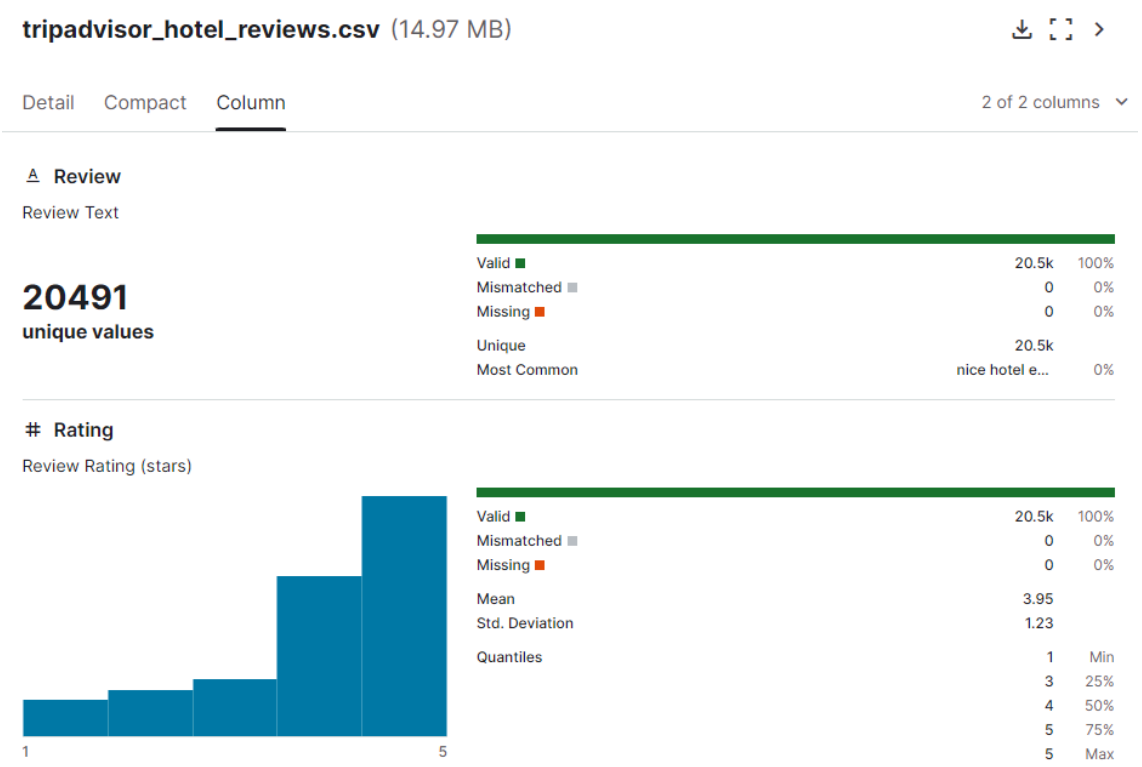

Figure 4: Dataset

(source)

3.3 Exploratory Data Analysis

In this very basic EDA is performed on the review i.e only the white spaces and stopwords are removed and the words are extracted from the review.

Pseudocode:

```
def preprocess_text(text):
  tokens = word_tokenize(text)
  # Removing the stopwords and punctuation
  stop_words = set(stopwords.words('english'))
  words = [word.lower() for word in tokens if word.isalpha() and word.lower() not in stop_words]
  return words
```

```
text="nice      parking ,    food was not good    "
sen=preprocess_text(text);
print(sen);
```

```
['nice', 'parking', 'food', 'good']
```

Figure 5: EDA Result Example

(source)

3.4 EXTRACTING MOST FREQUENT WORDS

For dynamic aspect classification we require some approach to select the aspects dynamically with the changing dataset(or addition of more review in the same dataset).
Here, the initial intuition we got is to choose dynamic aspects as the most frequent words of all the reviews in the document(dataset).

Problem with the initial intuition : Words like it,he,was,are,etc will also be counted as a choice for most frequent words.
Solution:We classify the words on parts of speech e.g. punctuation mark, determinant, adjective, noun,etc using NLTK library, and will select only the nouns as the possible contender for the aspects However, some nouns might exist that can never be an aspect.

Pseudo code :-

```
top_words = word_freq.most_common(20)
words, frequencies = zip(*top_words)
plt.figure(figsize=(10, 6))
plt.bar(words, frequencies, color=plt.cm.viridis(range(len(words))))
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Top 20 Most Frequent Words')
plt.xticks(rotation=45)
plt.show()
```

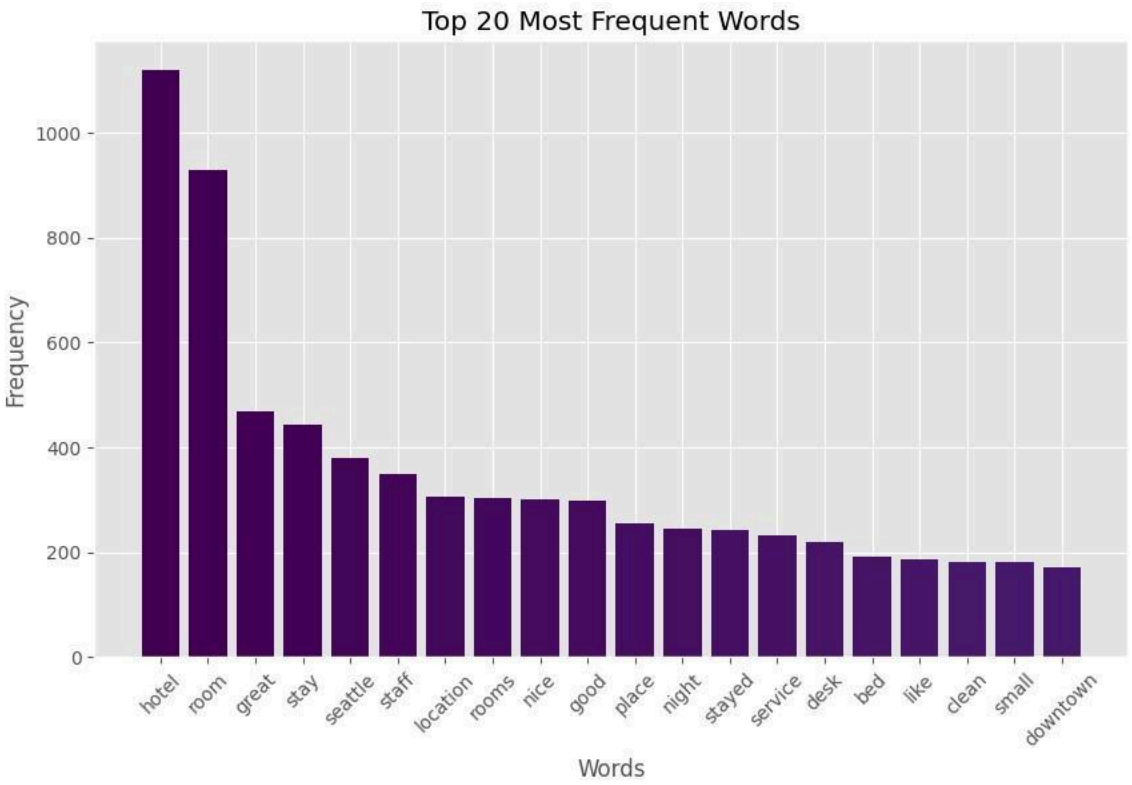

Figure 5: Most Frequent Words

(source)

Until here NLTK is not applied i.e all words fight as a contender for most frequent word.
NLTK will be applied in the dynamic aspect classifier function(**Section 3.9**).

3.5 BASIC NLTK

NLTK helps in classifying words in different parts of speech.

Pseudocode:

```
example = "the hotel is good , food is awesome, service is bad"
print(example)
tokens = nltk.word_tokenize(example)
tokens[:10]
tagged = nltk.pos_tag(tokens)
tagged[:10]
```

Output:

```
[('the', 'DT'),
 ('hotel', 'NN'),
 ('is', 'VBZ'),
 ('good', 'JJ'),
 (',', ','),
 ('food', 'NN'),
 ('is', 'VBZ'),
 ('awesome', 'JJ'),
 (',', ','),
 ('service', 'NN')]
```

Reference:https://www.geeksforgeeks.org/part-speech-tagging-stop-words-using-nltk-python/

3.6 VADER SENTIMENT SCORING

We are using NLTK's SentimentIntensityAnalyzer to analyze the sentiment of our text data. This analyzer gives sentiment scores for text, telling us if it's positive, neutral, or negative. First, we set up the SentimentIntensityAnalyzer as 'sia'. Then, we test it on three example sentences to show how it works. The 'sia.polarity_scores()' method gives sentiment polarity scores for each sentence. This step is really important because it helps us understand the sentiment in our text data. These scores can be used for different tasks like sentiment analysis by aspect, sentiment classification, or tracking sentiment trends. To analyze sentiment across all our data, we go through each review, figure out the sentiment scores, and keep the results in a dictionary named 'res'. Each review ID is linked to its sentiment scores. Next, we change these results into a 'DataFrame' called 'vaders', where each row shows a review with its sentiment scores and other info. By combining this sentiment scores DataFrame with our original dataset using review IDs, we enrich the dataset with sentiment details, making it easier to analyze sentiment trends. It's important to document this process so we know how sentiment analysis was done and how sentiment scores were added to our dataset. This makes sure that our sentiment analysis is transparent and reproducible, which helps us make informed decisions based on sentiment insights from our data.

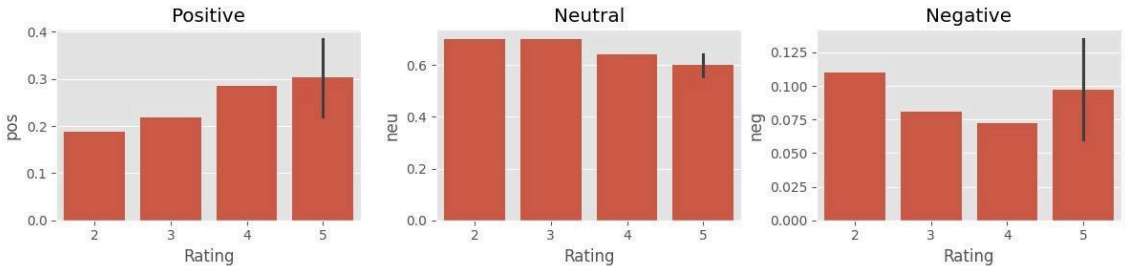

Figure 6: Plot of VADER results (source)

3.7 ROBERTA PRE-TRAINED MODEL

Here,we are using a pre-trained RoBERTa model fine-tuned for sentiment analysis. Firstly, we set up and load the pre-trained RoBERTa model, created for sequence classification and sentiment analysis, along with its tokenizer for text processing. Then, we show sentiment analysis using both VADER and the RoBERTa model on a sample sentence. We display the sentence and compute sentiment scores with VADER's polarity_scores() function. After that, we process the same sentence using the RoBERTa model. We encode the text with the RoBERTa tokenizer and run it through the model to get sentiment scores. The model output is a set of logits representing probabilities over sentiment classes (negative, neutral, positive), which we convert to probabilities using the softmax function. Lastly, we organize the sentiment scores from the RoBERTa model into a dictionary for easy interpretation and comparison, containing probabilities for each sentiment class: negative, neutral, and positive. This code showcases the use of both VADER and a pre-trained RoBERTa model for sentiment analysis, offering insights into sentiments conveyed in text data. By comparing results from both methods, we can assess their performance and suitability for our specific sentiment analysis task.

Pseudocode:

```
def polarity_scores_roberta(example):
   encoded_text = tokenizer(example, return_tensors='pt')
   output = model(**encoded_text)
   scores = output[0][0].detach().numpy()
   scores = softmax(scores)
   scores_dict = {
      'roberta_neg' : scores[0],
      'roberta_neu' : scores[1],
      'roberta_pos' : scores[2]
   }
   return scores_dict
```

3.8 STATIC ASPECTS CLASSIFICATION

Important aspects based on a hotel review are declared as aspects manually by us ,like parking,room,food,location and staff. Each has its own string for storing the review(or part of review).

Corresponding synonyms are manually stored in a dictionary,if any word matches the word from the dictionary then that sentence will be added to the review text of that key aspect.

Pseudo code :-

```
def aspect_classifier(text):
    aspects = {
        'parking':[
            "parking",
            "car parking",
            "parking lot",
            "parking space",
            "parking area",
            "parking garage",
            "parking spot",
            "vehicle parking",
            "automobile parking",
            "vehicle storage",
            "parking facility"
        ],
        'food':[
            "food",
            "cuisine",
            "restaurant",
            "edibles",
            "provisions",
            "diet",
            "meals"
        ],

        'room': [
            'apartments',
            'rooms',
            'room',
            'bed',
            'view',
            'pillows',
            'bath',
            'bedrooms',
            'bedroom',
            'apartment'
        ],

        'staff': [
            'staff',
            'service',
            'employees',
            'workers'
        ],

        'location': [
            'location',
            'shopping',
            'walking distance',
            'venue',
            'address',
            'site',
            'buildings'
        ]
    }
```

```
    category_sentences = {category: [] for category in aspects}

    sentences = text.replace(',', '.').split('.')
    for sentence in sentences:
        for category, keywords in aspects.items():
            if any(keyword in sentence.lower() for keyword in keywords):
                category_sentences[category].append(sentence)
    for category, sentences in category_sentences.items():
      category_sentences[category] = '. '.join(sentences)
    parking_review = ""
    food_review = ""
    room_review = ""
    staff_review = ""
    location_review = ""
    for category, text in category_sentences.items():
        if category == 'parking':
            parking_review = text
        elif category == 'food':
            food_review = text
        elif category == 'room':
            room_review = text
        elif category == 'staff':
            staff_review = text
        elif category == 'location':
            location_review = text

    return parking_review, food_review, room_review, staff_review, location_review
```

3.9 DYNAMIC ASPECTS CLASSIFICATION

Intuition with Pseudocode :-

1. Defined a function aspect_classifier_dynamic(text) that dynamically classifies aspects in a given text using NLTK and WordNet.
   a. Extract nouns from the input text using NLTK's part-of-speech tagging.
   b. If more than 5 nouns are found, select the first 5 nouns; otherwise, use all nouns found.
   c. For each noun, find its synonyms using WordNet.
   d. Create a dictionary where each noun is mapped to its synonyms.
   e. Initialize an empty dictionary called 'category_sentences' to store sentences related to each aspect.
   f. Split the input text into sentences.
   g. Iterate through each sentence:
      i. For each aspect (noun), check if any of its synonyms appear in the sentence. If so, add the sentence to the corresponding aspect's list in 'category_sentences'.
   h. Combine the sentences for each aspect into a single string.
   i. Return the reviews for each aspect.

Code:

```
tagged = nltk.pos_tag(words)
print(tagged,'\n')
nouns=[]
for word,typ in tagged:
  if typ=='NN':
```

```
    nouns.append(word)

if(len(nouns)>5):
  nouns=nouns[:5]
print(nouns,'\n')
```

2.Run the function polarity_scores_roberta which will calculate the sentiment scores using Roberta.

3. Initialized variables to store total positive, negative, and neutral scores for each aspect.

4. Iterate through each row in the dataset:
   a. Extract the review text and unique identifier.
   b. Use aspect_classifier_dynamic() to classify the review into different aspects and obtain the reviews for each aspect.
   c. For each aspect, calculate sentiment scores using polarity_scores_roberta().
   d. Update the total positive, negative, and neutral scores for each aspect.

Code:

```
def aspect_classifier_dynamic(text):
    aspects = {}
    for value in nouns:
      synonyms = get_synonyms(value)
      aspects[value]=synonyms

    print(aspects)

    category_sentences = {category: [] for category in aspects}

    sentences = text.replace(',', '.').split('.')

    for sentence in sentences:
        for category, keywords in aspects.items():
            if any(keyword in sentence.lower() for keyword in keywords):
                category_sentences[category].append(sentence)

    for category, sentences in category_sentences.items():
      category_sentences[category] = '. '.join(sentences)

    a1_review = ""
    a2_review = ""
    a3_review = ""
    a4_review = ""
    a5_review = ""

    for category, text in category_sentences.items():
        if category == nouns[0]:
            a1_review = text
        elif category == nouns[1]:
            a2_review = text
        elif category == nouns[2]:
            a3_review = text
        elif category == nouns[3]:
            a4_review = text
        elif category == nouns[4]:
            a5_review = text
```

return a1_review, a2_review, a3_review, a4_review, a5_review

5. Output the total positive, negative, and neutral scores for each aspect.

3.10 COMPARISON BETWEEN VADER AND ROBERTA

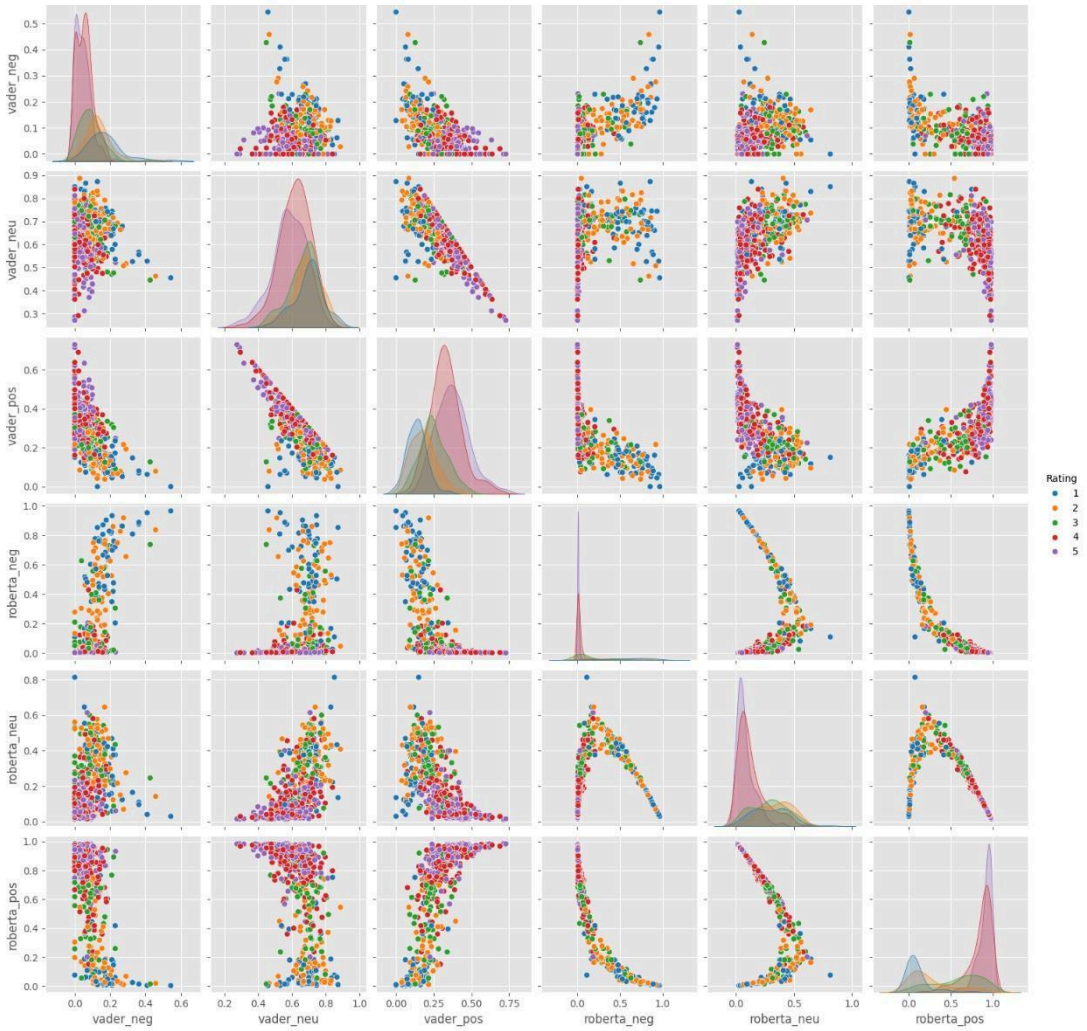

Figure 7: Comparison between VADER and ROBERTA

[(source)]

We did a comparison between VADER and ROBERTA model in specifying the whole review as sentiments.

ROBERTA produces results as a dictionary of then it is stored in three variables named :**roberta_neg (for negative tint),roberta_neu (for neutral tint), roberta_pos (telling the positive tint of the review)**

Pseudocode for ROBERTA:

def polarity_scores_roberta(example):

```
    encoded_text = tokenizer(example, return_tensors='pt')
    output = model(**encoded_text)
    scores = output[0][0].detach().numpy()
    scores = softmax(scores)
    scores_dict = {
        'roberta_neg' : scores[0],
        'roberta_neu' : scores[1],
        'roberta_pos' : scores[2]
    }
    return scores_dict
```

Similarly VADER's result is also stored in three variables: **vader_neg (for negative tint),vader_neu (for neutral tint), vader_pos (telling the positive tint of the review)**

Pseudocode for VADER:

```
vader_result = sia.polarity_scores(text)
vader_result_rename = {}
for key, value in vader_result.items():
    vader_result_rename[f"vader_{key}"] = value
```

These six variables are plotted against each other in a 6x6 matrix graph.
A single cell shows the classification of review based on the X-axis variable and Y-axis variable,

e.g Let's consider the first cell which has roberta_pos V/s vader_neg , the review along with corresponding ratings are divided into 5 different colors (see legend), then it is plotted on the graph with the variables values (suppose the vader_neg for the review was 0.7 and roberta_pos was 0.4 and rating corresponding to the review in the dataset is 2 , then it will be shown at (0.7,0.4) with an orange dot)

3.11 REVIEWING ANOMALIES

In this section, the anomalies of the results produced by the models are reviewed , the review's values of three variables and the corresponding rating is analysed.

Example: **Pseudocode for positively classified statement with a 1 star rating**

```
results_df.query('Rating ==  1') \
    .sort_values('roberta_pos', ascending=False)['Review'].values[0]

results_df.query('Rating == 1') \
    .sort_values('vader_pos', ascending=False)['Review'].values[0]
```

Pseudocode for negatively classified statement with a 5 star rating

```
results_df.query('Rating == 5') \
    .sort_values('roberta_neg', ascending=False)['Review'].values[0]

results_df.query('Rating == 5') \
    .sort_values('vader_neg', ascending=False)['Review'].values[0]
```

4 Result

The results include a detailed analysis of performance metrics, such as accuracy, precision, recall, and F1 score. Additionally, we discuss the classification of a review in different aspects and the corresponding roberta model values

	ID	vader_neg	vader_neu	vader_pos	vader_compound	roberta_neg	roberta_neu	roberta_pos	Review	Rating
0	1	0.072	0.643	0.285	0.9747	0.019479	0.104080	0.876442	nice hotel expensive parking got good deal sta...	4
1	2	0.110	0.701	0.189	0.9787	0.587673	0.358501	0.053826	ok nothing special charge diamond member hilto...	2
2	3	0.081	0.700	0.219	0.9889	0.289251	0.548734	0.162015	nice rooms not 4* experience hotel monaco seat...	3
3	4	0.060	0.555	0.385	0.9912	0.003897	0.033762	0.962341	unique, great stay, wonderful time hotel monac...	5
4	5	0.135	0.643	0.221	0.9797	0.073829	0.300111	0.626060	great stay great stay, went seahawk game aweso...	5

Figure 8: Results of VADER and Roberta compared

text="nice hotel expensive parking got good deal stay hotel anniversary, arrived late evening took advice previous reviews did valet parking, check quick easy, little disappointed non-existent view room room clean nice size, bed comfortable woke stiff neck high pillows, not soundproof like heard music room night morning loud bangs doors opening closing hear people talking hallway, maybe just noisy neighbors, aveda bath products nice, did not goldfish stay nice touch taken advantage staying longer, location great walking distance shopping, overall nice experience having pay 40 parking night"

vader_neg=0.072
vader_neu=0.643
vader_pos=0.285

roberta_neg=0.019479
roberta_neu=0.104080
roberta_pos=0.876442

Both VADER and roBERTa aligns themselves towards the positive sentiment but VADER shows more alignment towards the Neutral sentiment.Reason is the difference in working principle of these models.
VADER is a lexicon and rule-based sentiment analysis tool.It operates on the principle of using a pre-built list of words with associated sentiment scores (positive, negative, or neutral)

Cons which affects the scores of VADER:

- Relies heavily on the quality of the lexicon.
- Problematic with sarcasm and irony
- Not as effective for longer and more complex text.

These cons are handled by using the deep learning models such as RoBERTa,it is a transformer-based model, specifically a variant of BERT (Bidirectional Encoder Representations from Transformers). It is trained on a massive amount of text data using a masked language modeling objective and next sentence prediction task. RoBERTa considers the context of words, capturing the complex connections between them.
.

So observing these models we can infer that:

- **Approach:** VADER is lexicon and rule-based, while RoBERTa is based on deep learning and transformer architectures.
- **Interpretability:** VADER's decisions are easier to interpret, while RoBERTa's decisions are more difficult to interpret.
- **Performance:** RoBERTa generally achieves higher accuracy,but it requires more computational resources.

4.1 Performance Metrics

For ”parking” aspect: Accuracy: 0.8, Precision: 0.85, Recall: 0.8, F1 Score: 0.7809523809523808
For ”room” aspect: Accuracy: 0.6, Precision: 0.6, Recall: 0.6, F1 Score: 0.6
For ”food” aspect: Accuracy: 0.8, Precision: 0.64, Recall: 0.8, F1 Score: 0.7111111111111111
For ”staff” aspect: Accuracy: 0.8, Precision: 0.9, Recall: 0.8, F1 Score: 0.8
For ”location” aspect: Accuracy: 0.8, Precision: 1.0, Recall: 0.8, F1 Score: 0.8800000000000001

4.2 Confusion Matrix Analysis

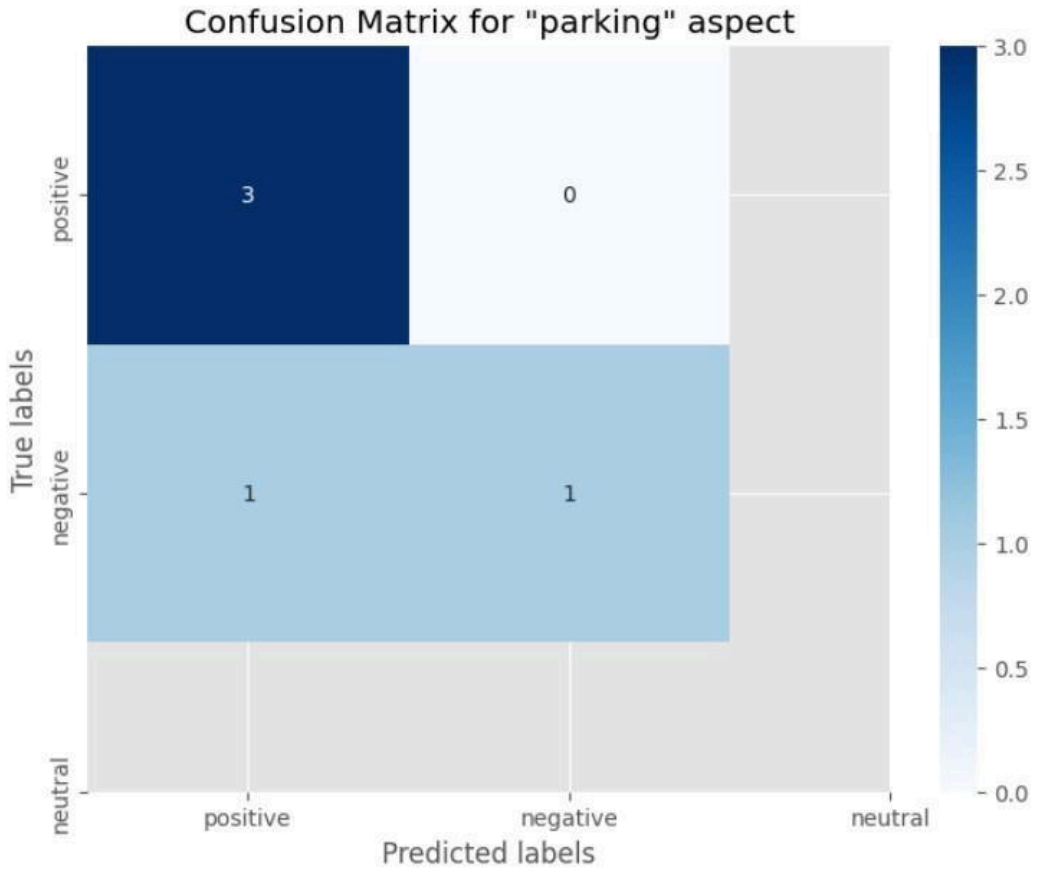

Figure 9: CONFUSION MATRIX FOR "parking" ASPECT (source)

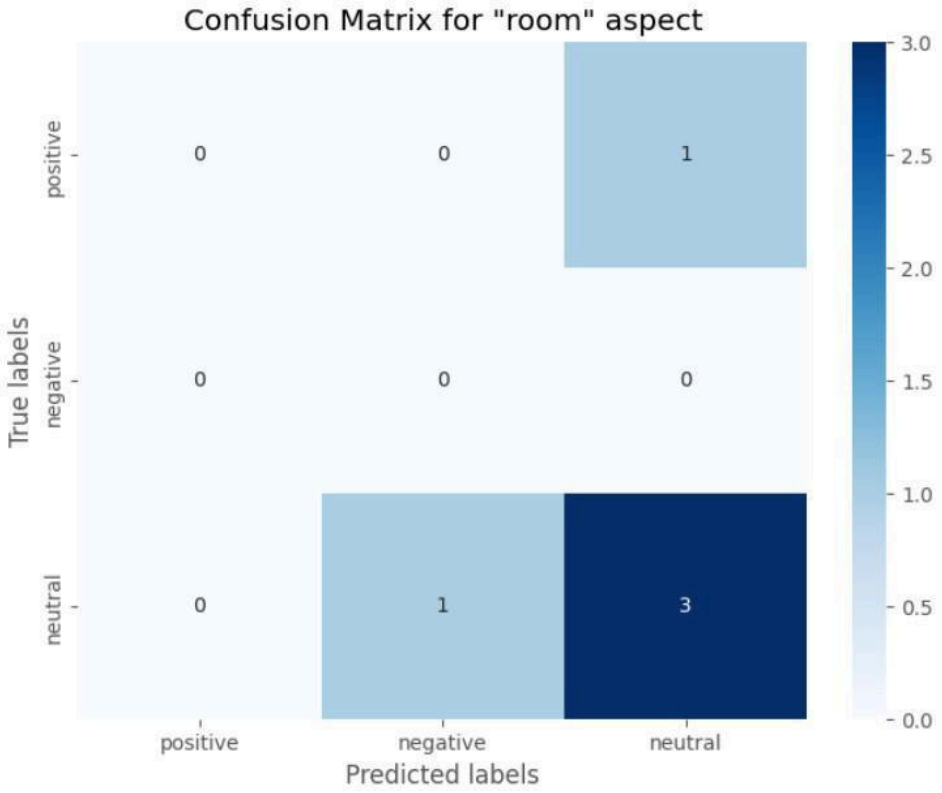

Figure 10: CONFUSION MATRIX FOR "room" ASPECT (source)

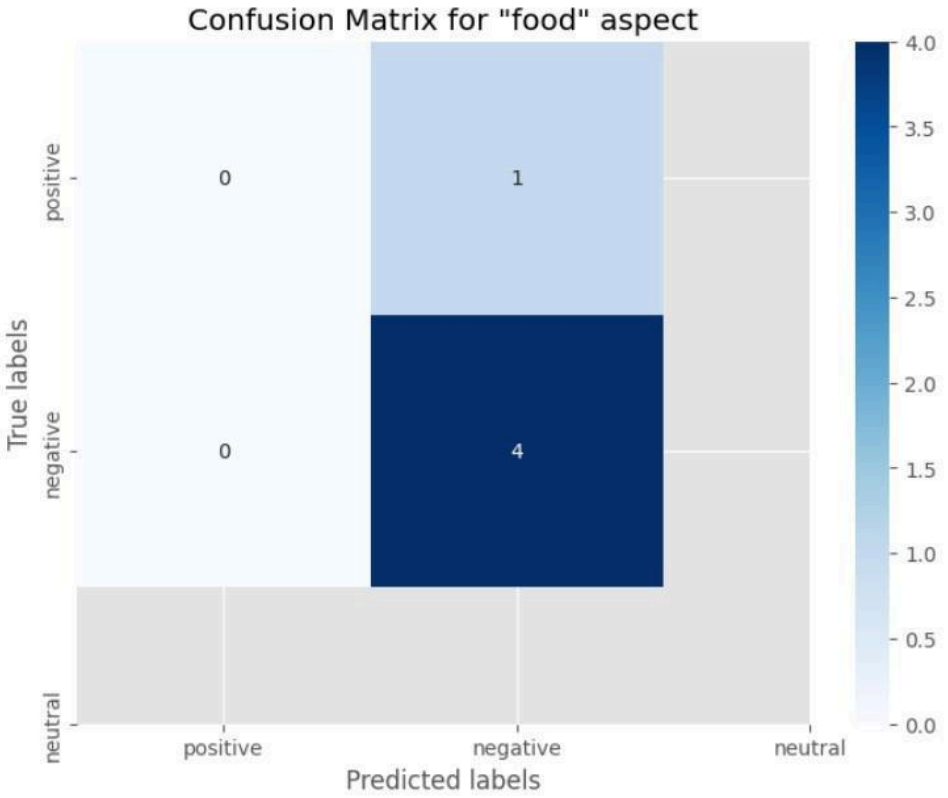

Figure 11: CONFUSION MATRIX FOR "food" ASPECT (source)

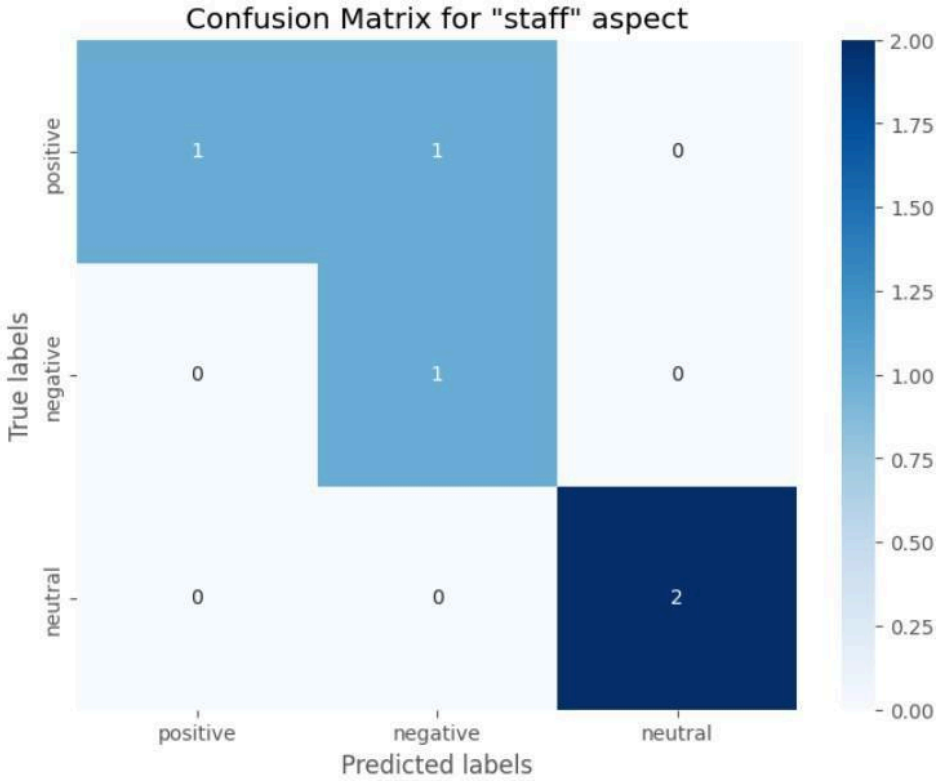

Figure 12: CONFUSION MATRIX FOR "staff" ASPECT (source)

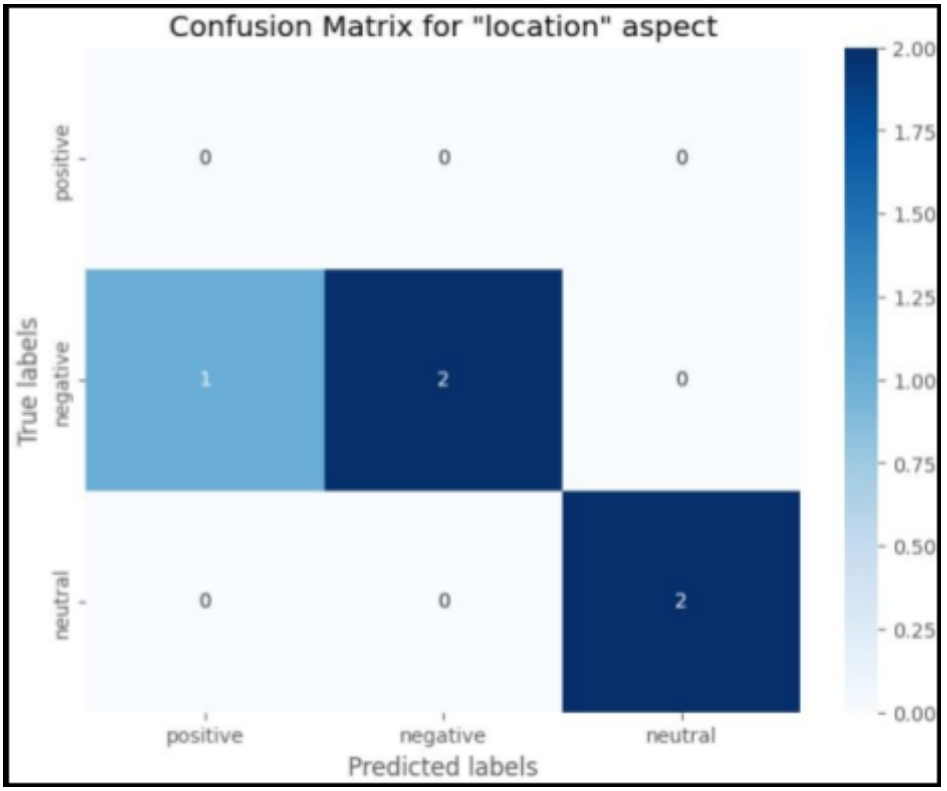

Figure 13: CONFUSION MATRIX FOR "location" ASPECT(source)

4.3 Discussion of Results

Few already known facts:

A high accuracy indicates that the model is making correct predictions overall, but it might not capture the performance of individual classes.

Precision is important when false positives are to be concerned more (e.g., medical diagnosis), whereas recall is important when the cost of false negatives is high (e.g., spam detection).
The F1 score provides a balance between precision and recall. It's useful when you want to compare models that have different precision-recall trade-offs.

On observing the metrics,deductions can be made easily.

Review:"nice hotel expensive parking got good deal stay hotel anniversary, arrived late evening took advice previous reviews did valet parking, check quick easy, little disappointed non-existent view room room clean nice size, bed comfortable woke stiff neck high pillows, not soundproof like heard music room night morning loud bangs doors opening closing hear people talking hallway, maybe just noisy neighbors, aveda bath products nice, did not goldfish stay nice touch taken advantage staying longer, location great walking distance shopping, overall nice experience having pay 40 parking night,"

Analysis of a review(as per the code's output):

Parking Review:
"Nice hotel, expensive parking got a good deal stay hotel anniversary. arrived late evening took advice previous reviews did valet parking. overall nice experience having pay 40 parking night "

{'roberta_neg': 0.0058923373, 'roberta_neu': 0.046260305, 'roberta_pos': 0.9478473}

Food Review:""
{'roberta_neg': 0.2582943, 'roberta_neu': 0.4512724, 'roberta_pos': 0.29043332}
Room Review:
"arrived late evening took advice previous reviews did valet parking. a little disappointed non-existent view room room clean nice size. The bed comfortably woke up with stiff neck high pillows. not soundproof like a heard music room night morning loud bangs doors opening closing hear people talking hallway. aveda bath products nice "

{'roberta_neg': 0.2505665, 'roberta_neu': 0.39840052, 'roberta_pos': 0.35103288}

Staff Review:""
{'roberta_neg': 0.2582943, 'roberta_neu': 0.4512724, 'roberta_pos': 0.29043332}

Location Review:
"location great walking distance shopping"

{'roberta_neg': 0.0030151908, 'roberta_neu': 0.08874555, 'roberta_pos': 0.9082393}

Whole Review Analysis:
{'roberta_neg': 0.019478872, 'roberta_neu': 0.10407953, 'roberta_pos': 0.87644154}

5 Conclusion

In conclusion, we performed Aspect Based Sentiment Analysis from the existing models(RoBERTa and VADER), and got satisfactory results also.

Initial intuition and Problems :

1)We started with thinking about how Aspects can be extracted from the review, the very first intuition was to statically define the aspects.

2)Next, the model which is to be used for sentiment analysis was to be chosen. Several sentiment analysis models exist, but we decided to move forward with VADER (lexicon based model) and RoBERTa (deep learning model).

3)A single review might contain reviews related to multiple aspects , so we created different strings for different aspects and created a static dictionary with key as aspect and values as the synonyms of the aspect to identify similar words related to the aspect.

4)Review is iterated sentence by sentence and then sentence is iterated word by word, if the word is found in the dictionary then that complete sentence is added to the key aspect's string. Break is not encountered yet, that is a single sentence might contain multiple words from the dictionary.So it will be added to multiple strings of the aspects.

5)Similarly the whole dataset is iterated review by review and each review sentence by sentence and each sentence as word by word.

6)Atlast, we have different strings containing sentences related to different static aspects

7)The models are run onto those strings and the polarity scores are added review by review.

8)All polarity scores are added and divided by the number of rows(to make the polarity score between 0 and 1)

9)But if an aspect is not present in the review then also it is added which reduces the overall average.

10)To solve this, we first checked for the presence of aspect in the review and models are run only if it is found ,otherwise it is not added in the polarity scores.Therefore, we created different n's for each aspect.

11)Thus new average for each aspect=total polarity scores/total number of rows having that aspect.

12)But these static aspects might lead to inaccurate results for new dataset or if new reviews are added in the same dataset. Thus we need dynamic aspect classification.

13)Frequency of occurrence is chosen as a factor which will help in the selection of aspects dynamically.

14)Different words are categorized based on the parts of speech (using NLTK) and only nouns are considered as a possible contender for the aspects.

15)So ,nouns with highest frequencies are chosen as the aspects for the current dataset.

16)If more reviews are added then the frequency of the words will change leading to new aspects.